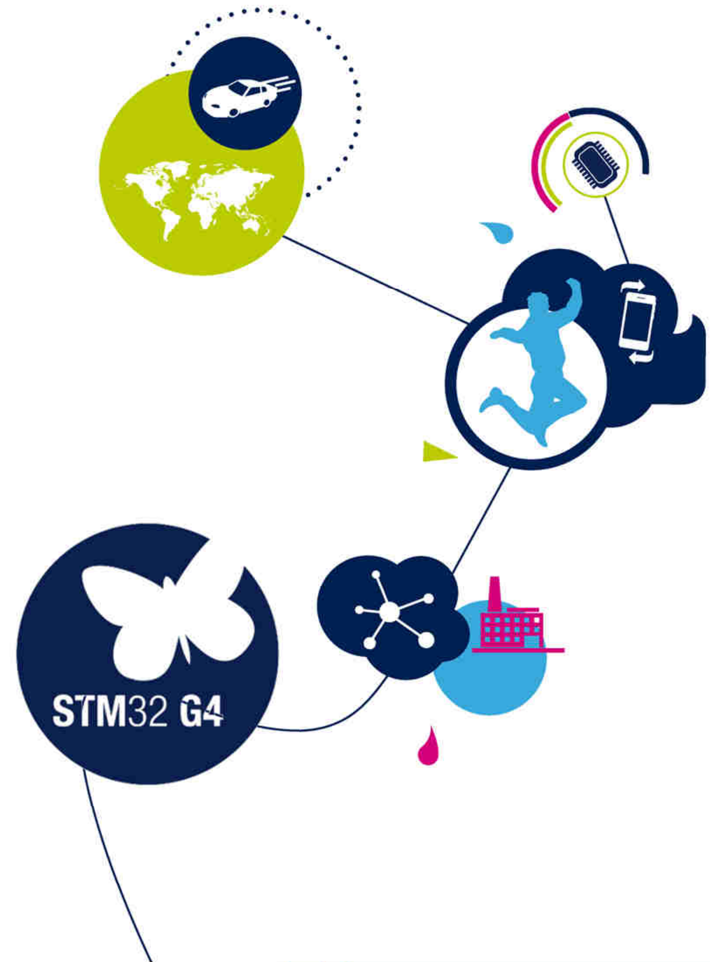
STM32G4 Mixed-signal MCU workshop

Learn to design with the new STM32G4 mixed-signal microcontroller



life.augmented

Technology Tour 2019

Vancouver, BC | September 24

STM32G4 mixed-signal MCU workshop

Agenda

1:00PM – 5:00PM	Introduction to the STM32G4 Series Hands On: STM32G474 Discovery Kit Demo Hands On: ADC Gain Compensation Hands On: DAC Signal Generation Hands On: Programmable Gain Amplifier Hands On: High Resolution Timer (HRTIM) Hands On: CORDIC Hardware Accelerator Hands On: Filter Math Accelerator (FMAC) Hands On: Dual Bank Flash Live Firmware Upgrade Hands On: CCM-SRAM Wrap Up

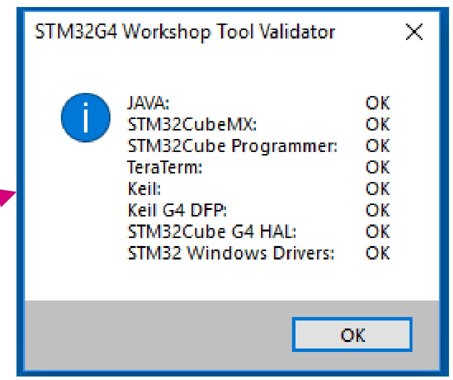
life.augmented

All set for the Hands-On?

Per your email invitation, you should have installed on your System:

- **STM32CubeMX:** v5.3.0 minimum
- **STM32CubeProgrammer:** v2.1.0 minimum
  If you installed **v2.1.0**, also execute
  “C:\Program Files (x86)\STMicroelectronics\STM32Cube\STM32CubeProgrammer\Drivers\stsw-link009_v3**dpinst_amd64.exe**”
- **STM32CubeG4:** v1.1.0 minimum
- **KEIL MDK-ARM:** v5.28a minimum
- **KEIL MDK-ARM LICENSE:** Pro Eval License Add On
- **Tera Term** or similar

Run the validator to check status



Please ask for help if you have any problems

STM32 General Usage Survey

Please complete this survey prior to the start of the workshop

Use your phone to scan the QR code or type the link into your browser.

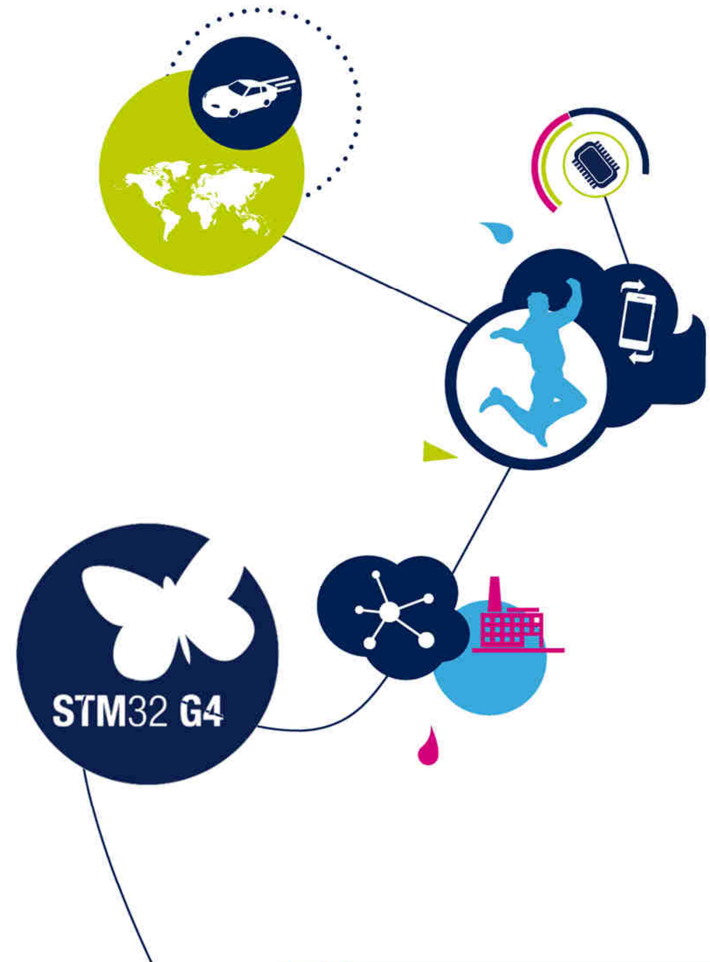
https://www.surveymonkey.com/r/68NZNQR

We Greatly Value Your Feedback - Thank you!

life.augmented

STM32G4 Mixed-signal MCU workshop

Learn to design with the new STM32G4 mixed-signal microcontroller



life.augmented

Technology Tour 2019

Vancouver, BC | September 24

STM32 portfolio

17 product series / More than 50 product lines

	Cortex-M0 Cortex-M0+	Cortex-M3	Cortex-M33	Cortex-M4	Cortex-M7	Dual Core MCU Cortex®-M7 & Cortex®-M4	Multi Core MPU Dual Core Cortex®-A7 & Cortex®-M4
High-performance		STM32 F2		STM32 F4	STM32 F7, STM32 H7	STM32 H7 ●	STM32 MP1 ●
Mainstream	STM32 F0, STM32 G0	STM32 F1		STM32 F3, STM32 G4 (NEW)			
Ultra-low-power	STM32 L0	STM32 L1	STM32 L5	STM32 L4, STM32 L4+			
Wireless				STM32 WB ●			

arm

Note ●: Cortex-M0+ Radio Co-processor, ● : Cortex-M4 Co-processor

life.augmented

STM32 Rolling Longevity Commitment

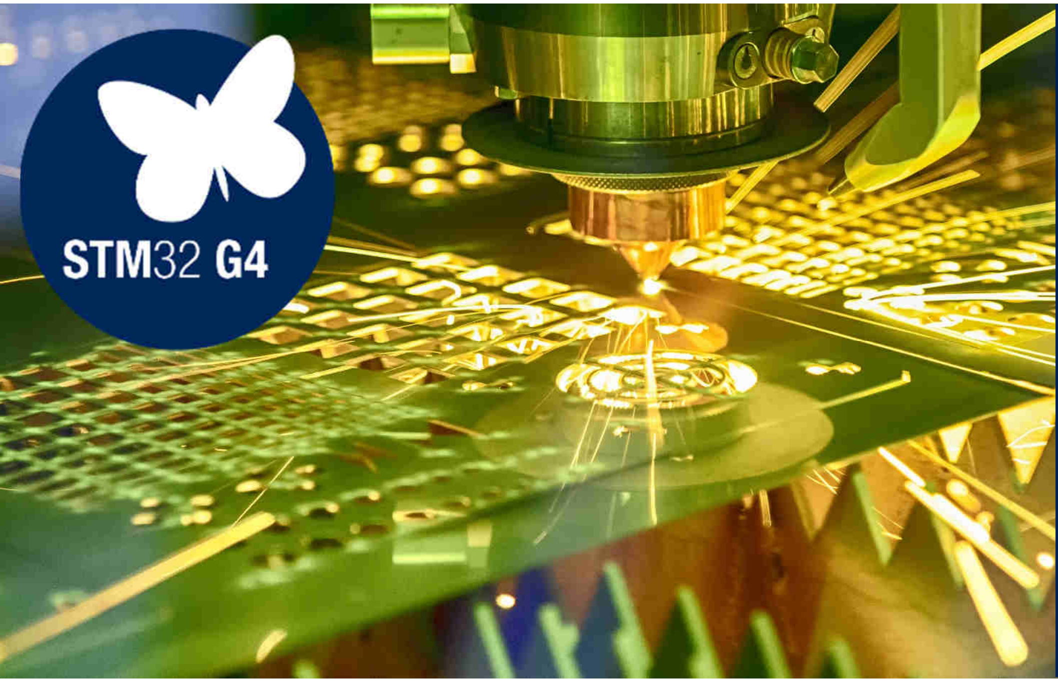
Longevity commitment is renewed every year

starting January 1st 2019
➔ Until 2029

STM32F1	(launched in **2007**)	**22 years** of commitment
STM32L1	(launched in **2009**)	**20 years** of commitment
STM32F2	(launched in **2010**)	**19 years** of commitment
STM32F4	(launched in **2011**)	**18 years** of commitment
STM32F0	(launched in **2012**)	**17 years** of commitment
STM32F3	(launched in **2012**)	**17 years** of commitment
STM32L0	(launched in **2013**)	**16 years** of commitment
STM32F7	(launched in **2014**)	**15 years** of commitment
STM32L4	(launched in **2015**)	**14 years** of commitment
STM32L4+	(launched in **2016**)	**13 years** of commitment
STM32H7	(launched in **2017**)	**12 years** of commitment
STM32WB	(launched in **2018**)	**11 years** of commitment
STM32G0	(launched in **2018**)	**11 years** of commitment
STM32G4	(launched in **2019**)	**10 years** of commitment

STM32G4 Series

Ideal for applications requiring MCU with advanced and rich analog peripherals



- Control applications (Motor Control…)
- Industrial equipment
- Instrumentation and Measurement
- Digital Power
    - Digital SMPS (switch mode power supply)
    - PFC (power factor correction)

STM32 G4

STM32G4 Series – Key Messages

Performance

- Arm® Cortex®-M4 at 170 MHz
- 213 DMIPS and 550 CoreMark® results
- Better dynamic power consumption (163µA/MHz)
- ART Accelerator™ (dynamic cache)
- Mathematical accelerators
- CCM-SRAM Routine Booster (static cache)

Rich Integrated Analog and Digital

- Op-Amps (Built-in gain), DACs, Comparators
- 12-bit ADCs 4Msps with hardware oversampling
- CAN-FD (flexible data rate – 8Msps bit rate)
- High resolution timer (184 ps)
- USB type-C Power Delivery3.0
- 1% RC accuracy [-5°..90°C], 2% full T° range

Safety and security focus

- Dual Bank Flash with ECC (error code correction)
- Securable Memory Area
- Hardware encryption AES-256
- SIL, Class-B
- SRAM with Parity bit

Secure Live Upgrade

Functional safety design packages

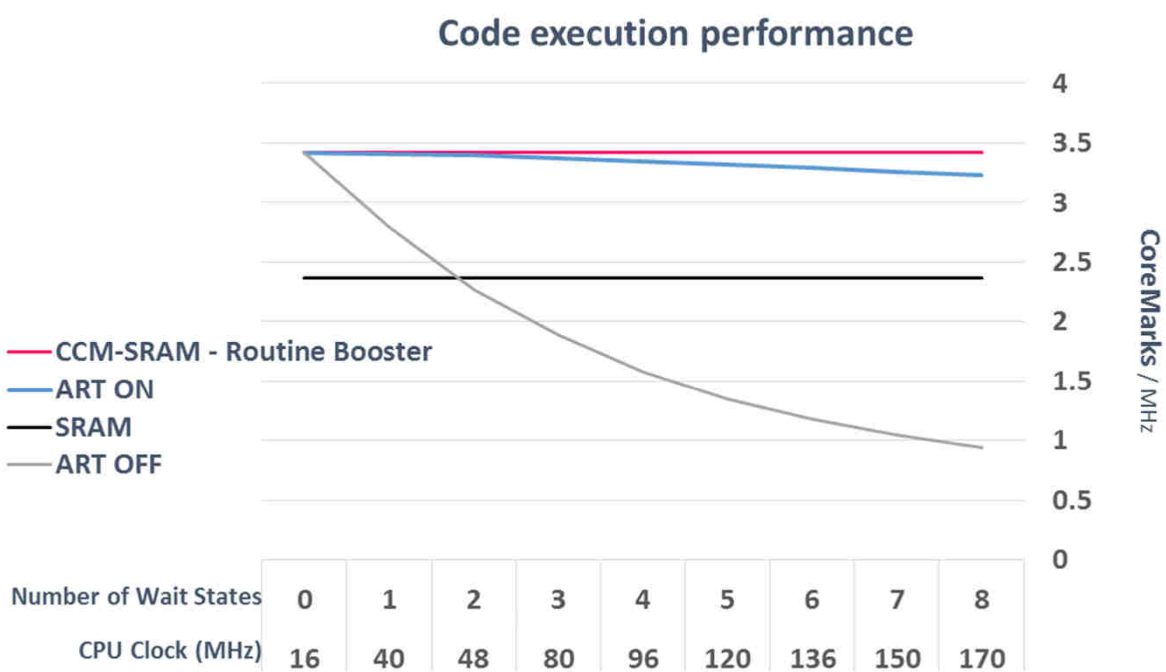
Complete portfolio

- Complements existing STM32F3 Series portfolio
- From -40°C up to 85 or 125°C devices
- From 32- up to 128-pin
- From 32KB to 512KB Flash

life.augmented

Greater Performance

Pure 170 MHz CPU performance (Arm® Cortex®-M4) with 3 accelerators



Arm Cortex-M4 with **FPU**

Up to 170 MHz CPU frequency

Up to 213 DMIPS and 550 CoreMark® results

3 different **HW accelerators**:

- **ART accelerator** (~dynamic cache)
  → Full code acceleration (average)
- **Routine Booster** CCM-SRAM (~static cache) → determinism preserved
- **Mathematical** (CORDIC + FMAC) NEW

NEW Mathematical Accelerators

Function acceleration and CPU offload

1. CORDIC (Trigo)

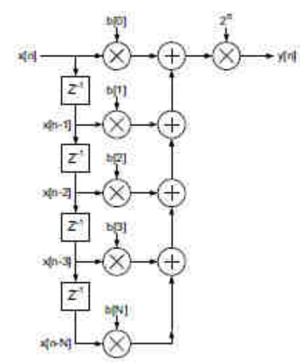
- Very helpful for Field Oriented Motor Control method (FOC)

- Vector rotation (polar to rectangular): Sin, Cos
- Vector translation (rectangular to polar): Atan2, Modulus
- Sinh, Cosh, Exp
- Atan, Atanh
- Square root
- Ln

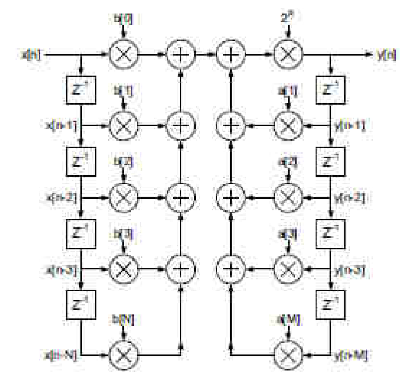
2. Filter Math Accelerator (FMAC)

- Can be used to create
  - 3p3z Compensator (→ Digital power)
  - Sigma Delta modulator
  - Noise Shaper

FIR filter



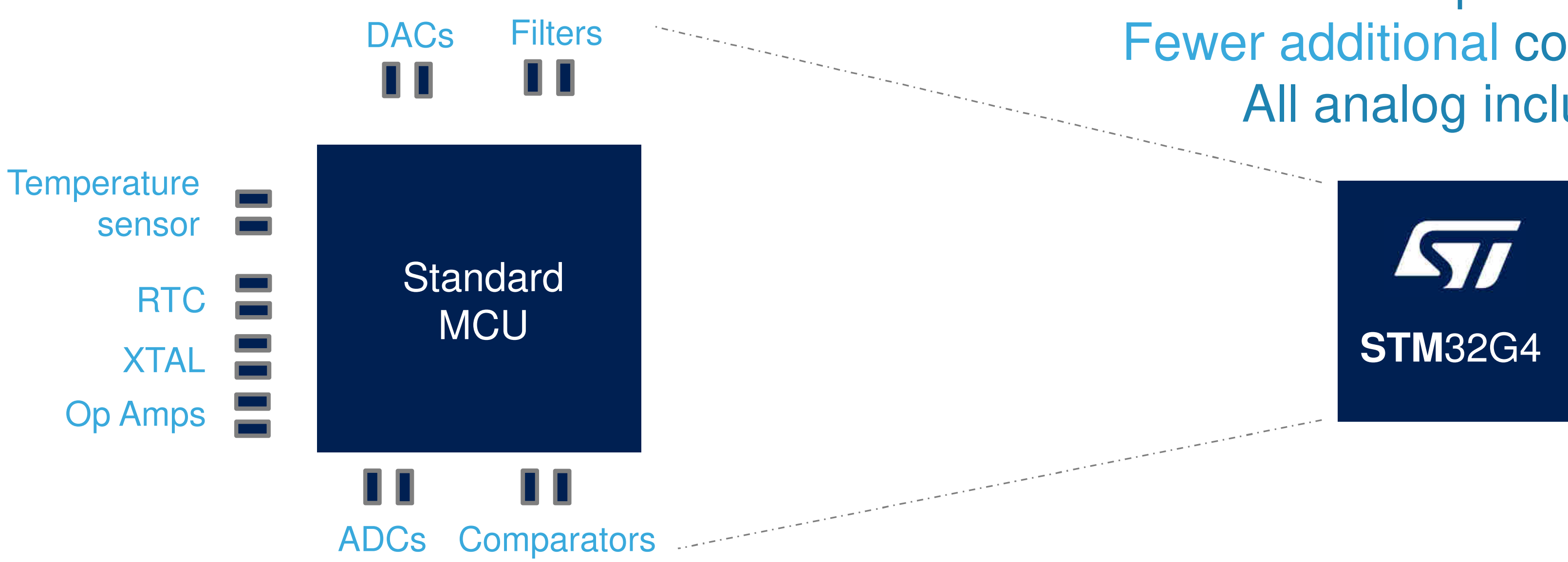
IIR filter



Reducing PCB Size and BOM Cost

System-on-Chip – All-in-one solution

Smaller package
Fewer additional components
All analog included



Project cost $$$

Project cost $

life.augmented

Rich, Advanced Analog

Mixed-signal SoC for wide variety of applications

ADC **(up to 5)**	**Values**
Topology	**SAR 12-bit** + HW oversampling → **16-bit**
Sampling rate	Up to **4 Msps**
Input	Single-ended and differential
Offset and Gain compensation	Auto calibration to reduce gain and offset

DAC **(up to 7)**	**Values**
Sampling rate	**15 Msps** (internal) 1Msps (from buffered output)
Settling time	16ns

Op-Amp **(up to 6)**	**Values**
GBW	**13 MHz**
Slew rate	**45 V/µs**
Offset	3mV over full T° range 1.5mV @ 25°C
PGA Gain (accuracy)	2, 4, 8, 16, -1,-3,-7,-15 **(1%)** 32, 64, -31,-63 (2%)

Comparator **(up to 7)**	**Values**
Power supply	1.62 .. 3.6V
Propagation delay	**16.7ns**
Offset	-6 .. +2 mV
Hysteresis	8 steps: 0, 9, 18, 27, 36, 45, 54, 63 mV

STM32 G4

Key Features for Targeted Applications

Motor Control

Home appliances, E-bikes, Air Conditioning

- Fast CPU 170MHz
- Mathematical accelerator (Cordic)
- Advanced Motor Control timers
- Fast comparators
- 4Msps ADC-12bit + HW oversampling
- Op-Amp with built-in gain (PGA)
- DAC-12bit
- 1% RC accuracy
  (UART communication w/o external Xtal)

High-End Consumer

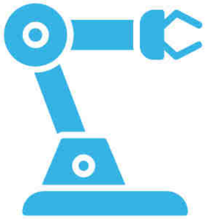
Rechargeable devices, drones, toys

- Low-thickness, small form-factor
- Low consumption in run mode ~ 160µA/MHz
- Embedded analog
- SAI (Sound Audio Interface)
- USB type-C Power Delivery 3.0

Industrial devices Measurements

Industrial equipment

- Fast CPU 170MHz
- Mathematical accelerator (Cordic)
- High temperature 125°C
- CAN FD support
- SPI, USART, I²C
- Advanced timers
- Real Time Clock with backup registers
- Dual bank flash for **live** upgrade
- AES & security



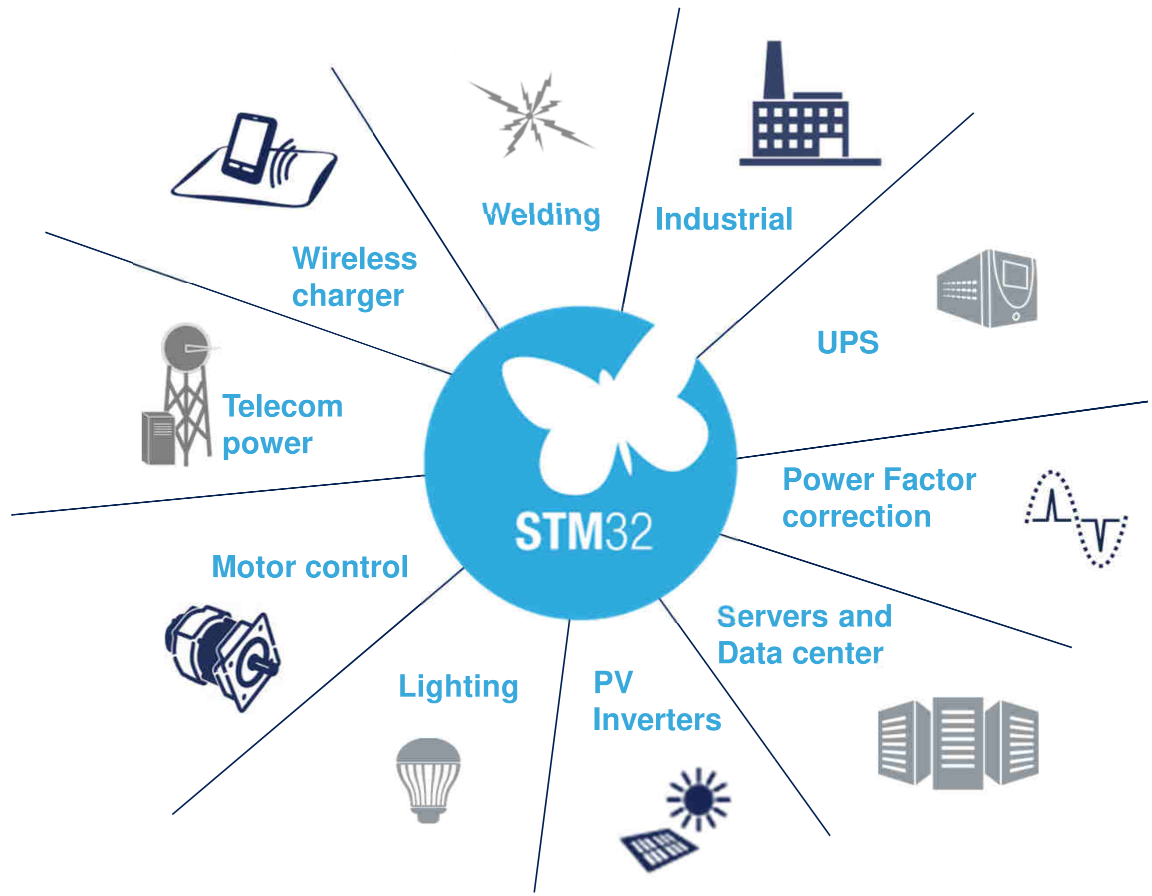
Digital Power

Servers, Telecom, EV Charging station

- Fast CPU 170 MHz
- Mathematical accelerator (Filtering)
- 12ch High Resolution timer (184ps)
- 4Msps ADC-12bit + HW oversampling
- Fast comparators (17ns)
- Embedded analog
- Dual bank flash for **live** upgrade
- AES & security
- FMAC for 3p3z compensation

Ease Digital Power Conversion

Enhance your digital power solutions using the STM32G4's full features High Resolution Timer (HRTIM)

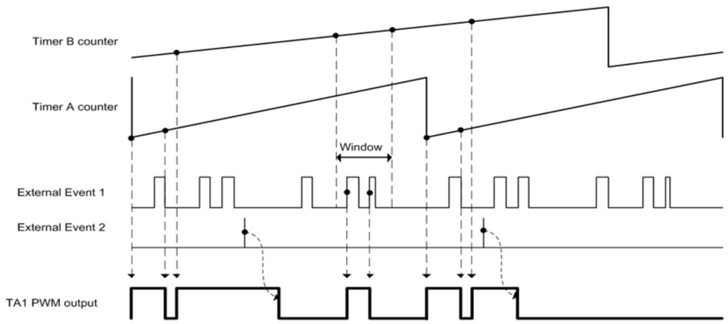


life.augmented

HRTimer – Not only High Resolution…

High resolution PWM

- 12 channels with 184ps resolution on frequency and duty cycle
- 184ps is equivalent to 5.4GHz timer clock



Flexible PWM generation

- 7x independent time base to create various shape of PWM
- 6x complementary pair PWM outputs
- Up to 32 set/reset transition per PWM period thx to the built-in crossbar
- Master/Slave configuration for multi phase converter

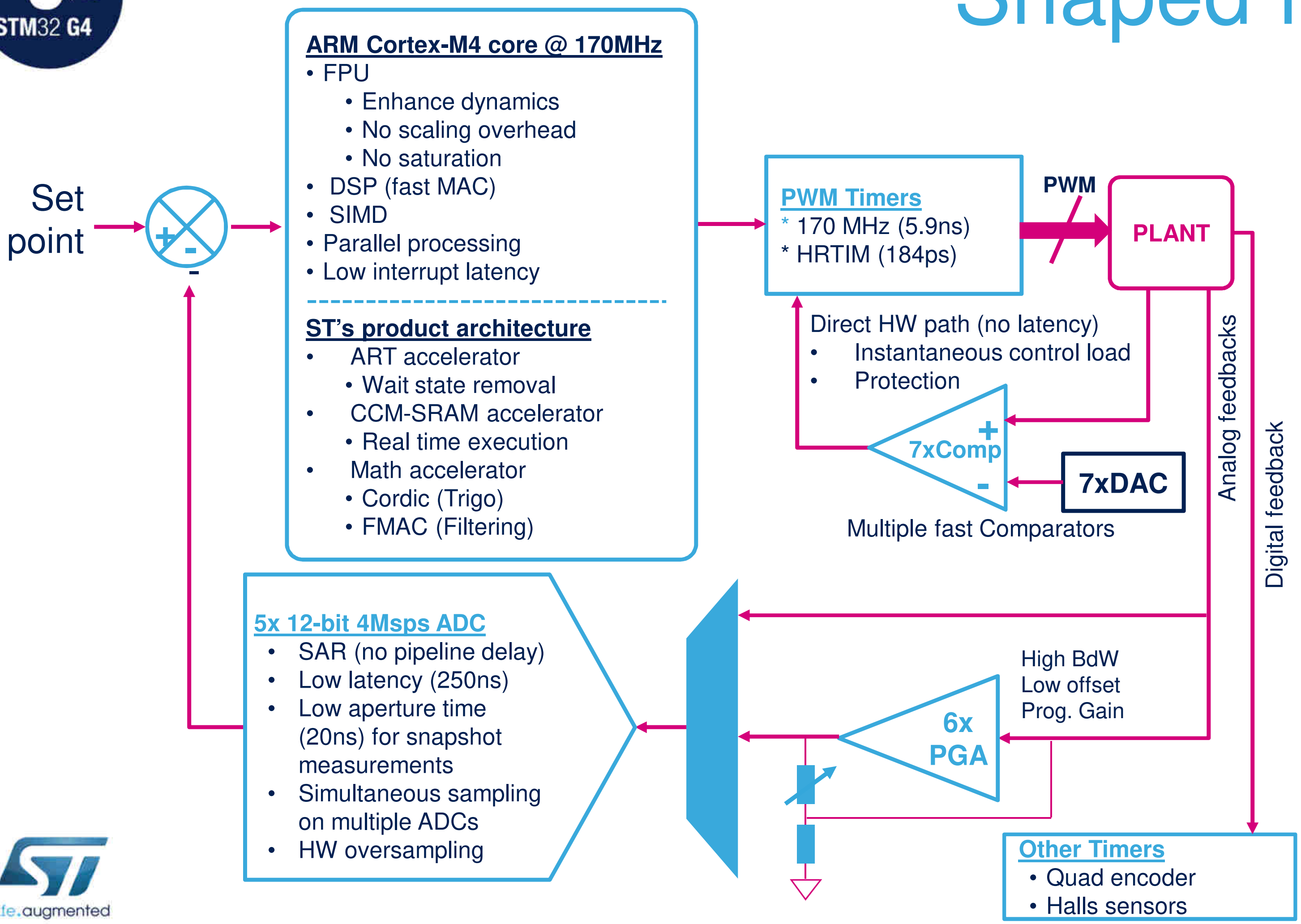
Multiple Event handler

- 6x Digital and Analog fault input
- 10x Events cycle to cycle current control or PWM restart (constant Ton/Toff)
- Blanking, windowing and digital filter

12 independent channels

- Any topology supported from 1x 12 PWM (triple interleaved LLC (servers application) up to 12x1 PWM (multiple independent buck converters (lighting)

Shaped for Control

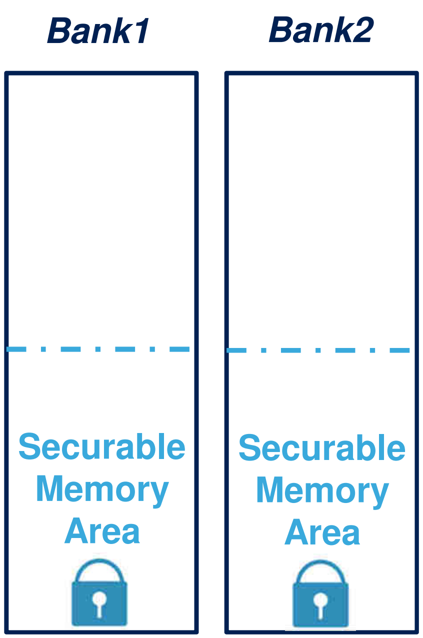


Easy to use the Analog and Digital resources thanks to high peripherals interconnect and flexible bus matrix

Greater Security

Integrated security features, ready for tomorrow’s needs

User Flash



STM32Trust: a complete toolset
an ecosystem for embedded security

STM32 Trust

https://www.st.com/stm32trust

Securable Memory Area:
- Configurable size
- Can be secured once exiting
  - No more access nor debug possible
- Good fit to store critical data
  - Critical routines
  - Keys

	Securable user memory	AES TRNG	PCROP	MPU	Readout protection	CRC	Write Protection
Secure firmware install (SFI)	●	●			●		
Secure Firmware upgrade (SFU)	●	●			●		●
Mutual Distrustful			●				
Firmware IP protection			●				
Secret key storage	●				●		
Secured communication		●				●	
Authentication	●	●			●		
Task cloisoning				●			

life.augmented

Dynamic Efficiency Modes

When Mainstream MCU Series meets low-power requirements

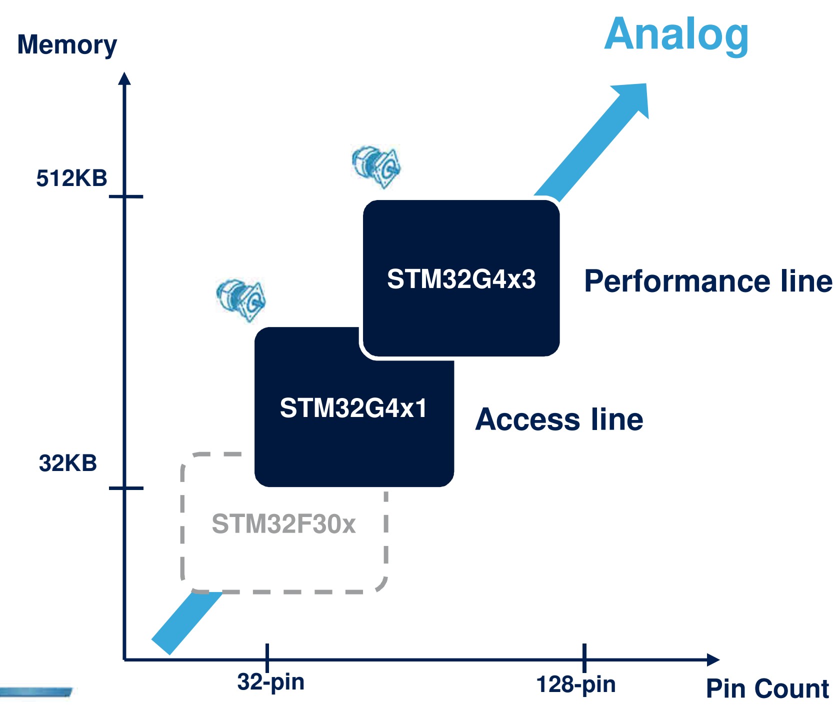
Wake-up time	Mode	Consumption	
	VBAT*	**7nA / 720 nA**	Tamper: few I/Os, RTC
268 µs	**SHUTDOWN***	**43nA / 565 nA**	Wake-up sources: reset pin, few I/Os, RTC
30 µs	**STANDBY***	**130nA / 885 nA**	Wake-up sources: + BOR, IWDG
9.5 µs	**STOP 1 (full retention)**	**80 µA**	Wake-up sources: + all I/Os, PVD, COMPs, LPUART, LPTIM, I²C, UART, USB
11 cycles	**SLEEP at 26 MHz**	**37 µA / MHz**	Wake-up sources: any interrupt or event
	RUN (Range1) at 150 MHz from Flash	**163 µA / MHz**	
	RUN (Range1 boost) at 170 MHz from Flash	**173 µA / MHz**	

Conditions: 25°C, V_{DD} = 3V

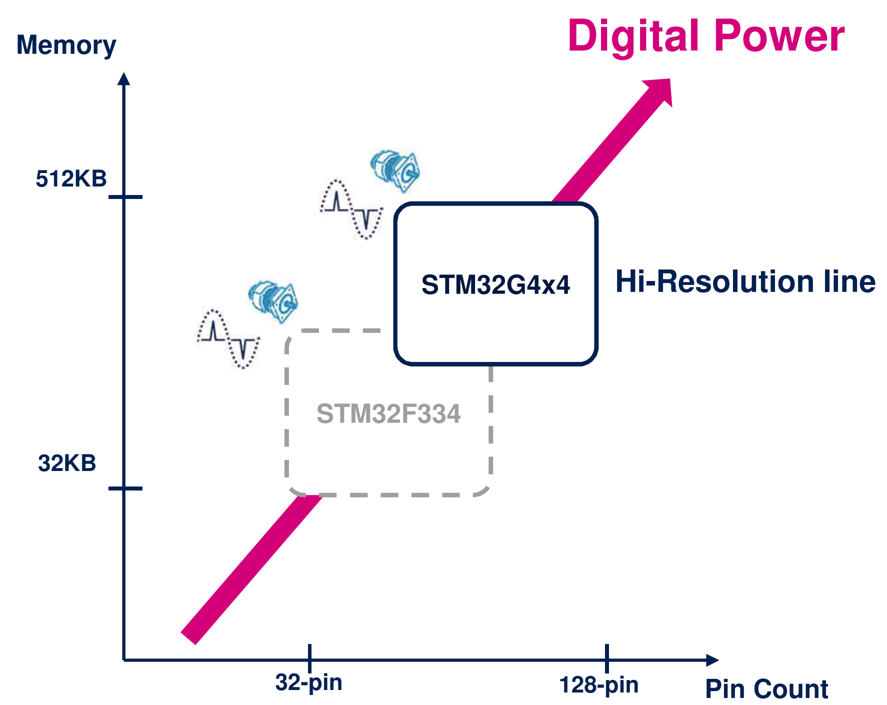
Note : * without RTC / with RTC

STM32G4 Products Lines

General Purpose



Applications Specific



Extensive & Innovative Peripheral Set

STM32 G4

No compromise on what matters

Unit parameters	STM32G474 Hi-Resolution line	STM32G473 Performance line	STM32G431 Access line
Core, frequency	Arm Cortex-M4, 170 MHz		
Flash (max)	512 Kbytes (2x256KB dual bank)		128 Kbytes single bank
RAM (up to)	96 Kbytes		22 Kbytes
CCM –SRAM (code-SRAM)	32 Kbytes		10 Kbytes
12-bit ADC SAR	4x 12-bit 4 MSPS		2x 12-bit 4 MSPS
Comparator	7		4
Op amp with 4 built-in gain values with 1% accuracy	6		3
12-bit DAC	7		4
Motor Control timer	3x (170 MHz)		2x (170 MHz)
CAN-FD	3x		1x
12 channel Hi-resolution Timer	1x	-	-
Power supply	1.72 to 3.6 V		

life.augmented

STM32 G4

STM32G47x

High Resolution and Performance lines [128KB .. 512KB]

- **32-bit Arm Cortex-M4 core with FPU**
- **ART + CCM-SRAM + Mathematic Accelerators**
- **Dual Bank Flash with ECC**
- **SRAM with Parity bit**
- **+/- 1% internal clock**
- **1.72 to 3.6V power supply**
- **Up to 125°C**

Connectivity

- 4x SPI, 4x I²C, 6x UxART
- 1x USB 2.0 FS, 1x USB-C PD3.0 (+PHY)
- 3x CAN-FD
- 2x I²S half duplex, SAI

External interface

- FSMC 8-/16-bit (TFT-LCD, SRAM, NOR, NAND)
- Quad SPI

Accelerators

- ART Accelerator™
- 32-Kbyte CCM-SRAM

Math Accelerators

- Cordic (trigo...) Filtering

Arm® Cortex®-M4
Up to 170 MHz
213 DMIPS

- Floating Point Unit
- Memory Protection Unit
- Embedded Trace Macrocell
- 16-channel DMA + MUX
- Up to 2x 256-Kbyte Flash memory / ECC Dual Bank
- 96-Kbyte SRAM

Timers

- 5x 16-bit timers
- 2x 16-bit basic timers
- 3x 16-bit advanced motor control timers
- 2x 32-bit timers
- 1x 16-bit LP timer
- 1x HR timer (D-Power) 12-channel w/ 184ps (A. delay line)

Analog

- 5x 12-bit ADC w/ HW overspl
- 7x Comparators
- 7x DAC (3x buff + 4x non-buff)
- 6x Op-Amp (PGA)
- 1x temperature sensor
- Internal voltage reference

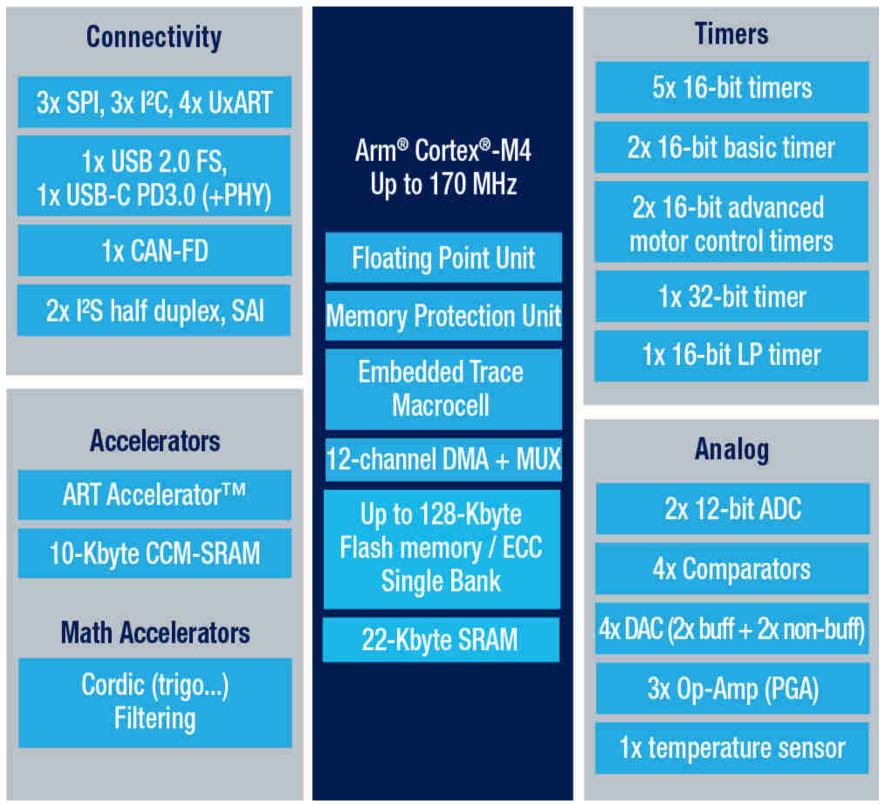
- **High resolution timer**
- **3x Advanced Motor Control timers**
- **Rich Advanced Analog**
- **3x CAN Flexible Data rate**
- **USB-C Power Delivery3.0**
- **Advanced Security and Safety features**
- **Robustness: highest level 5 / FTB/ESD - IEC 61000-4-4**

life.augmented

STM32G43x

Access line [32KB .. 128KB] and up to 512KB in H1-2020 !

- **32-bit Arm Cortex-M4 core with FPU**
- **ART + CCM-SRAM + Mathematic Accelerators**
- **Single Bank Flash with ECC**
- **SRAM with Parity bit**
- **+/- 1% internal clock**
- **1.72 to 3.6V power supply**
- **Up to 125°C**



- **2x Advanced Motor Control timers**
- **Rich Advanced Analog**
- **CAN Flexible Data rate**
- **USB-C Power Delivery3.0**
- **Advanced Security and Safety features**
- **Robustness: highest level 5 / FTB/ESD - IEC 61000-4-4**

life.augmented

STM32 G4

STM32G4 Portfolio

Flash memory / RAM size (bytes)	32-pin LQFP/QFN	48-pin LQFP/QFN	64-pin LQFP/WLCSP	80-pin LQFP/WLCSP	100-pin LQFP/BGA	128-pin LQFP
512 K / 128 K		STM32G484CE	STM32G484RE	STM32G484ME	STM32G484VE	STM32G484QE
		STM32G474CE	STM32G474RE	STM32G474ME	STM32G474VE	STM32G474QE
		STM32G483CE	STM32G483RE	STM32G483ME	STM32G483VE	STM32G483QE
		STM32G473CE	STM32G473RE	STM32G473ME	STM32G473VE	STM32G473QE
		STM32G491CE	STM32G491RE	STM32G491ME	STM32G491VE	
256 K / 128 K		STM32G474CC	STM32G474RC	STM32G474MC	STM32G474VC	STM32G474QC
		STM32G473CC	STM32G473RC	STM32G473MC	STM32G473VC	STM32G473QC
		STM32G491CC	STM32G491RC	STM32G491MC	STM32G491VC	
128 K / 128 K		STM32G474CB	STM32G474RB	STM32G474MB	STM32G474VB	STM32G474QB
		STM32G473CB	STM32G473RB	STM32G473MB	STM32G473VB	STM32G473QB
128 K / 32 K	STM32G441KB	STM32G441CB	STM32G441RB	STM32G441MB	STM32G441VB	
	STM32G431KB	STM32G431CB	STM32G431RB	STM32G431MB	STM32G431VB	
64 K / 32 K	STM32G431K8	STM32G431C8	STM32G431R8	STM32G431M8	STM32G431V8	
32 K / 32 K	STM32G431K6	STM32G431C6	STM32G431R6	STM32G431M6	STM32G431V6	

Pin count

Legend: Crypto AES-256 (dark blue: STM32G484xx, STM32G483xx, STM32G441xx) · Available in H1 2020 (grey: STM32G491xx)

life.augmented

Broad Portfolio

Portfolio extended to support budget applications efficiently

More memory and pin counts

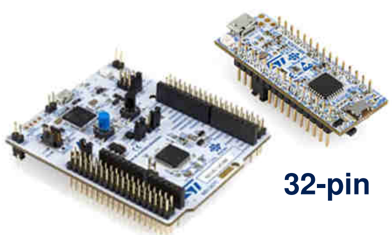
Flash memory (bytes)	32-pin LQFP QFN	48-pin LQFP QFN	64-pin LQFP BGA WLCSP	80-pin LQFP WLCSP	100-pin LQFP BGA	121-pin BGA	128-pin LQFP
512 K		✓	✓	✓	✓	✓	✓
256 K		✓	✓	✓	✓	✓	✓
128 K	✓	✓	✓	✓	✓	✓	✓
64 K	✓	✓	✓	✓	✓		
32 K	✓	✓	✓	✓	✓		

More packages

Note: new packages in STM32 portfolio

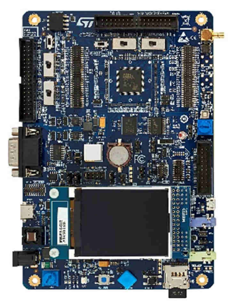
STM32G4 Hardware Solutions

Accelerate evaluation, prototyping and design



32-pin

64-pin



STM32 Nucleo

Flexible prototyping

- NUCLEO-G431RB
- NUCLEO-G474RE
- NUCLEO-G431KB

Evaluation boards

Full feature STM32G4 evaluation

- STM32G484E-EVAL
- STM32G474E-EVAL
- STM32G474E-EVAL1

Motor Control Pack

Full feature for Motor Control and Analog

- P-NUCLEO-IHM03

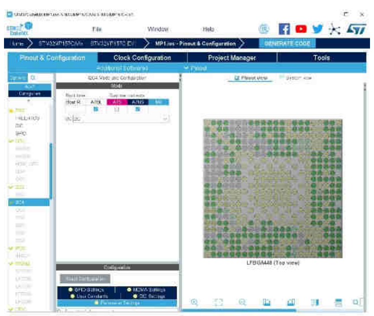
Discovery kits

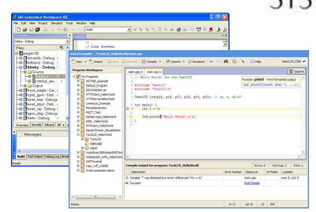
Key feature prototyping

- B-G474E-DPOW1
- B-G431B-ESC1

STM32G4 Software Tools

Complete support of Arm Cortex-M ecosystem





All-in-one STM32 programming tool
Multi-mode, user-friendly

STM32CubeMX

STM32CubeMX

- Configure and generate Code
- Conflicts solver

IDEs Compile and Debug

Flexible Solutions

- Partners IDE, like IAR and Keil
- Free IDE based on Eclipse, like STM32CubeIDE*

STM32 Programming Tool

STM32CubeProgrammer

- Flash and/or system memory
- GUI or command line interface

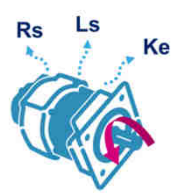
* SW examples will be available in Q4 19

Dedicated Ecosystems

Motor Control

- **Complete ecosystem** (HW boards, SW Development Kit (SDK), docs and trainings)
  - X-CUBE-MCSDK (v5.4)
    - Motor Control FW library based on STM32Cube HAL and LL
    - Motor control workbench: Graphical configurator of the motor control library linked with STM32CubeMx
  - P-NUCLEO-IHM03: Motor Control Nucleo pack
    - NUCLEO-G431RB Nucleo-64
    - X-NUCLEO-IHM16M1 motor driver expansion board
    - Low Voltage motor

- **State of the art algorithms** (FOC, 6-step, sensorless…)
- **Motor Profiler**: Plug and spin your motor within less than one minute

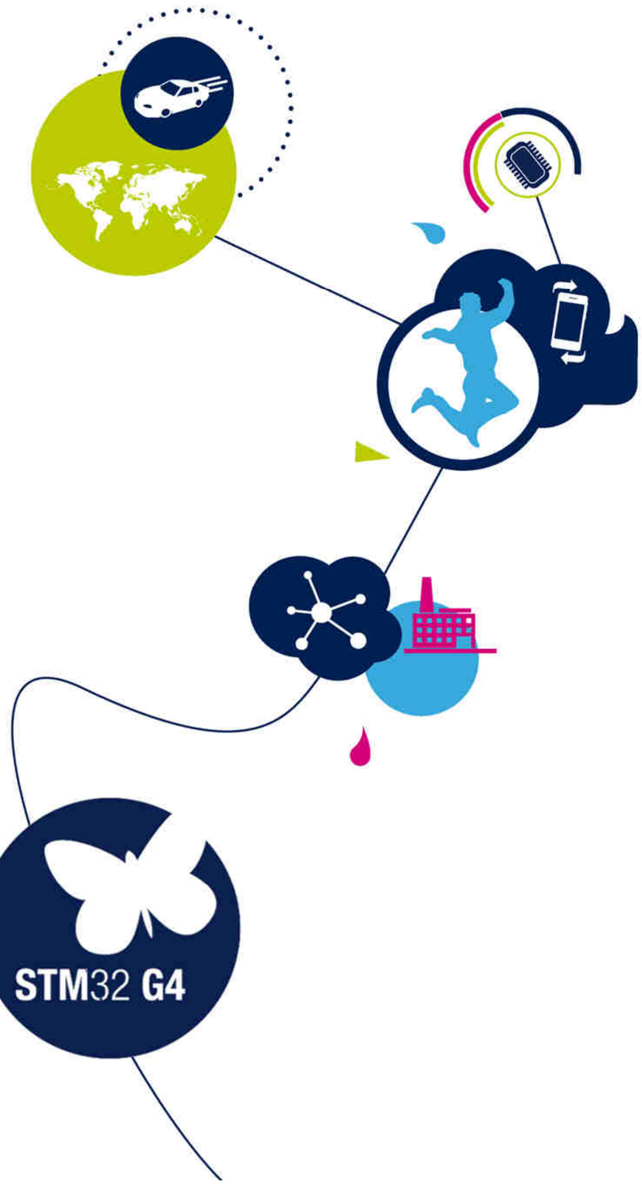


Digital Power

- **Complete ecosystem** (HW boards, FW examples, SW tools, docs and trainings)
- **Dedicated HRTIM Cook Book - AN4539:** How to operate the Hi-Resolution timer in different topology
- **Digital Power training** (PSU and PFC) – based on STM32 G4 series – done in collaboration with Biricha (from Q4 2019)

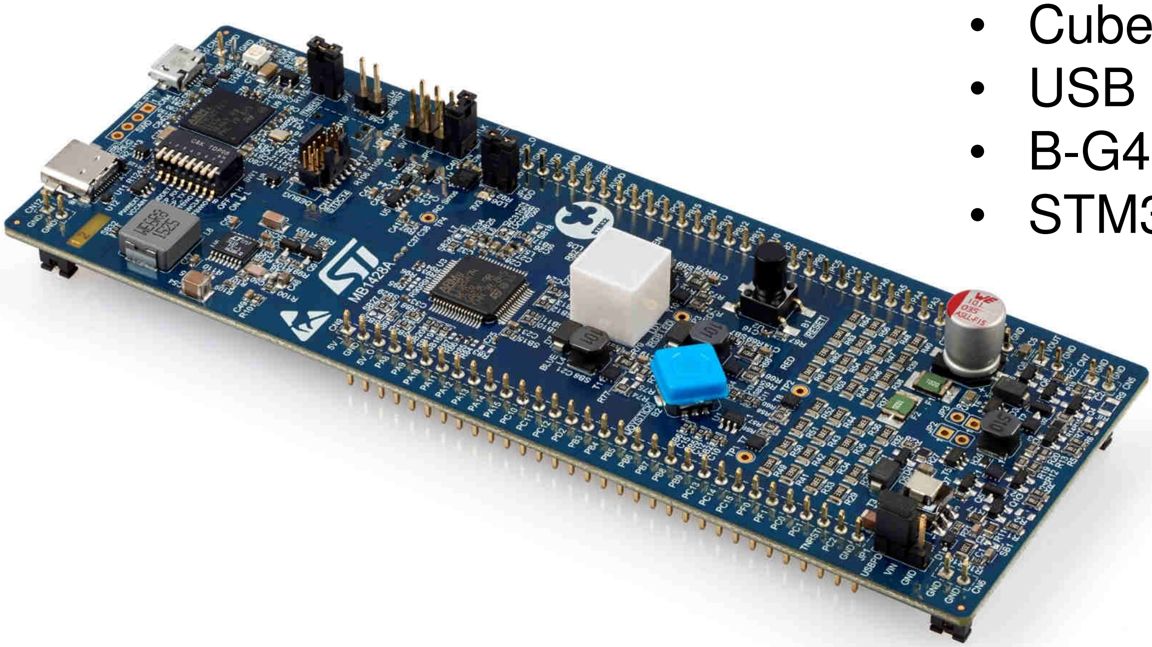
STM32G4 mixed-signal MCU workshop

Hands On:
STM32G4 Discovery Kit Demo



life.augmented

Systems Check



- Cube Programmer
- USB cable
- B-G474E-DPOW1 Discovery Board
- STM32G4 Family Cube Library

B-G474E-DPOW1

Jumpers – 1 of 2

Do not attach USB cable here

CN2
USB Type-C™ receptacle

All switches Down

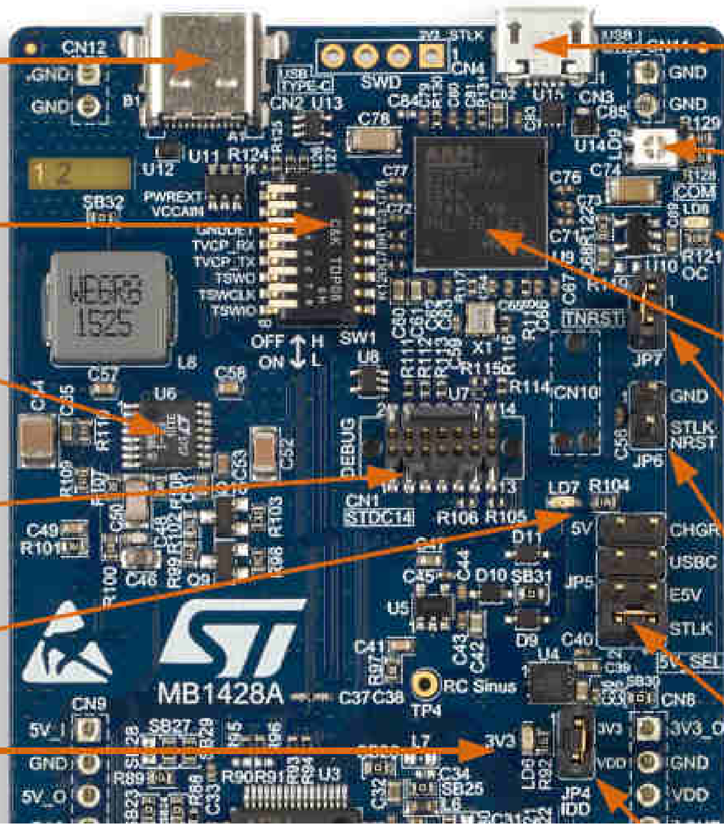
SW1
Octal mechanical switch

U6
LTC3114-1

CN1
STDC14 debug connector

LD7
5V LED

LD6
3V3 LED



CN3
USB micro-B
ST-LINK receptacle

Attach USB cable here

LD9
COM LED

LD8
overcurrent LED

U9
STLINK-V3E MCU

JP7
TNRST

INSTALLED

JP6
STLK NRST

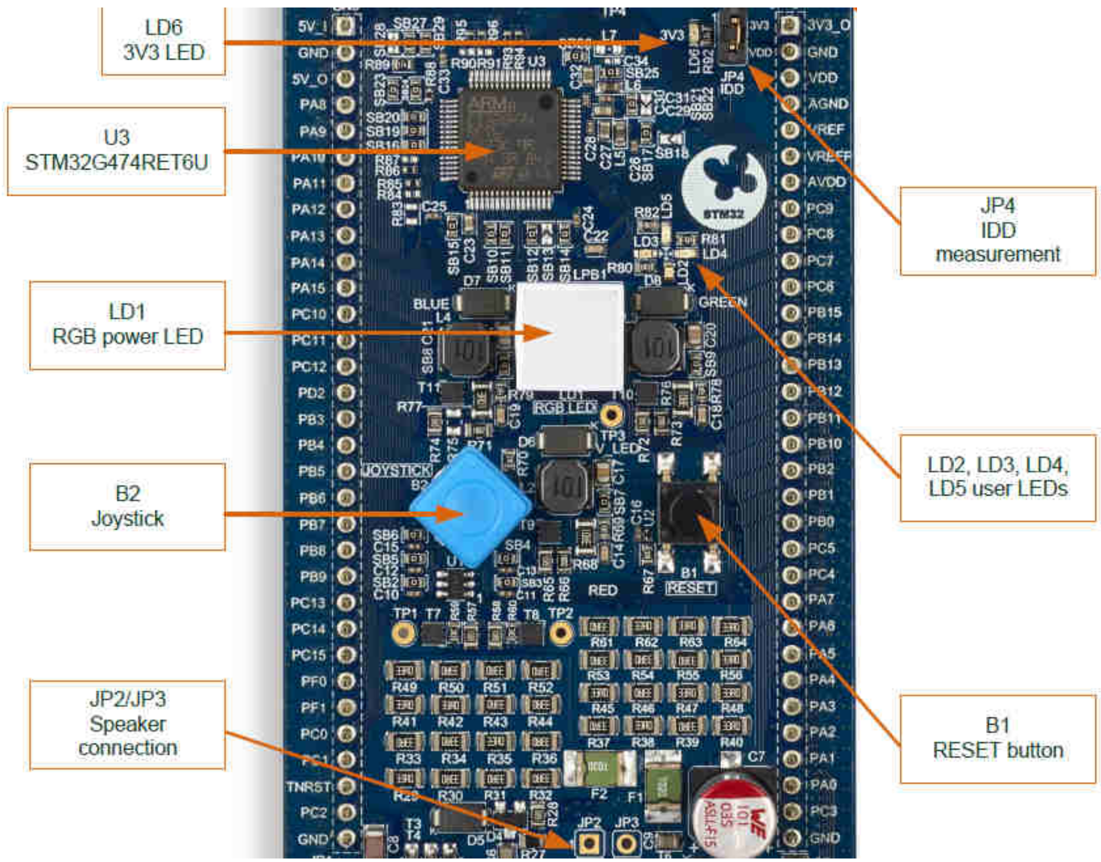
OPEN

JP5
5V_SELECTION

INSTALLED on STLK

B-G474E-DPOW1

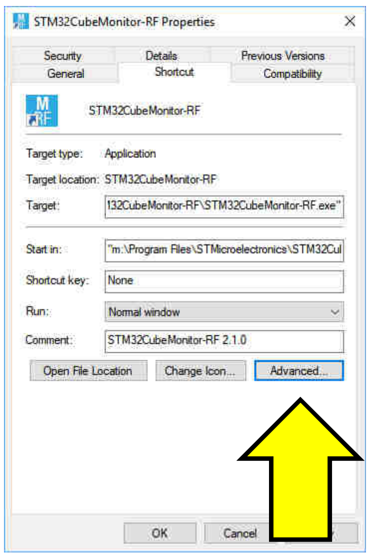
Jumpers – 2 of 2

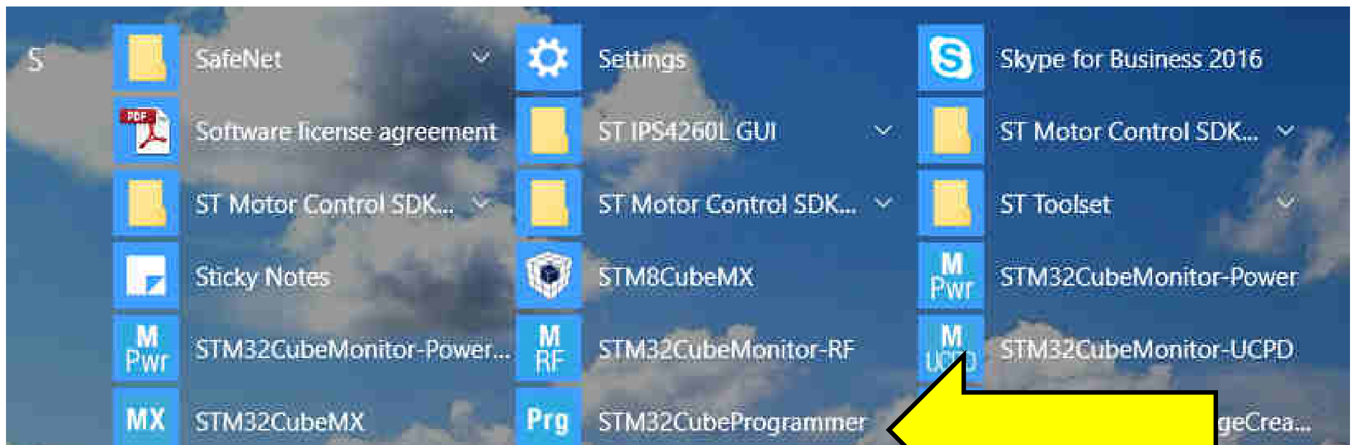


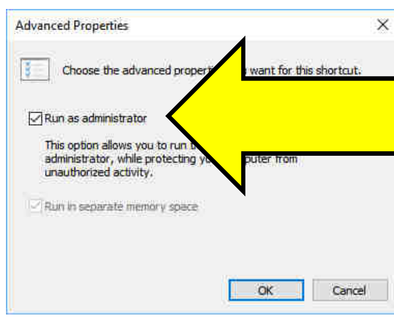
INSTALLED

CubeProgrammer

Start up

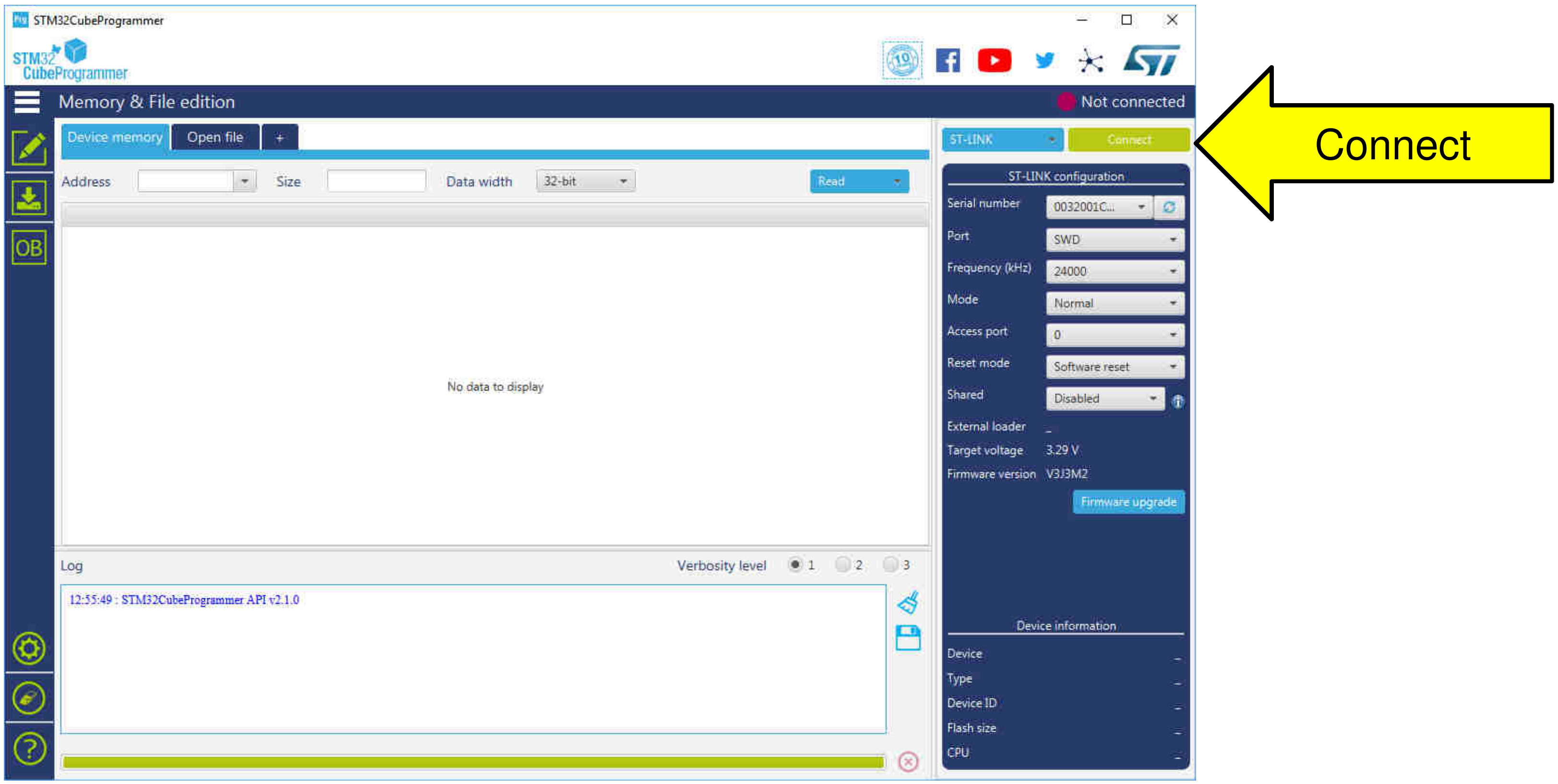






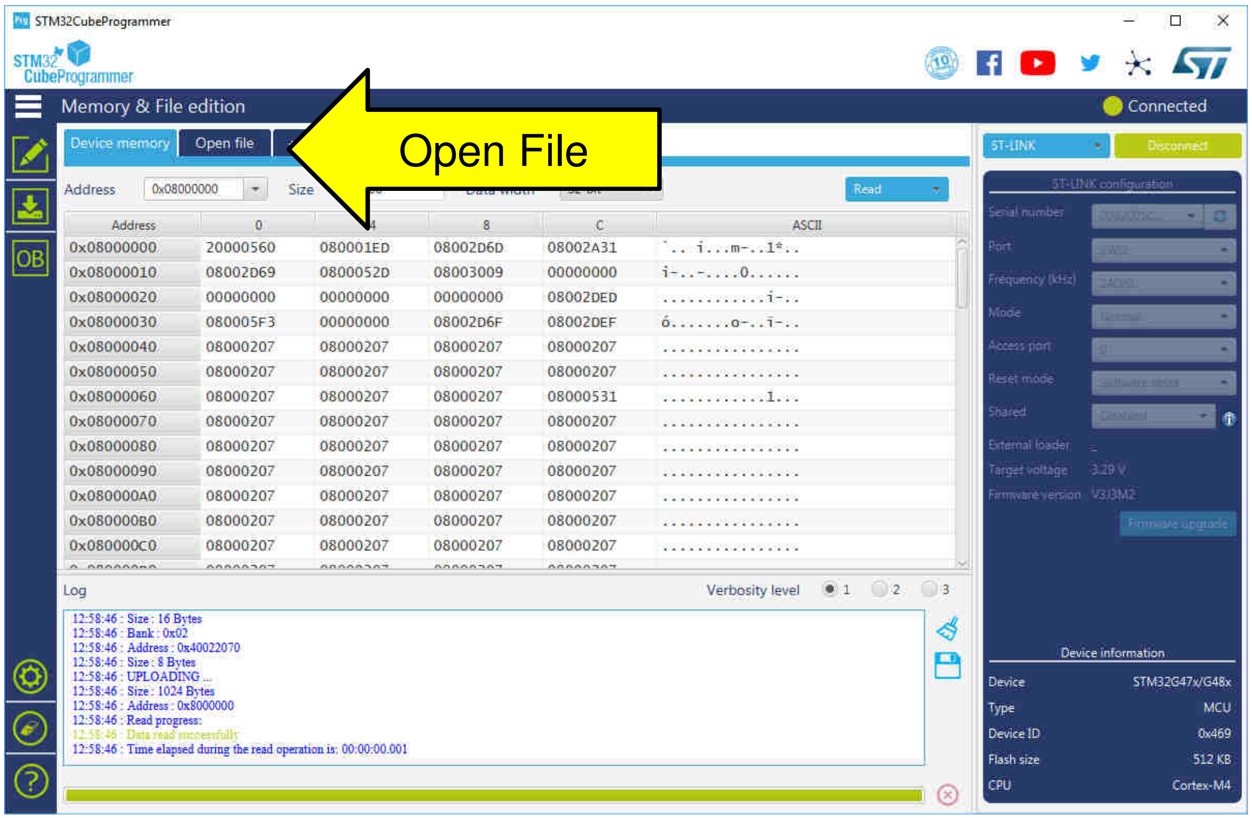
CubeProgrammer

Splash Page - Connect



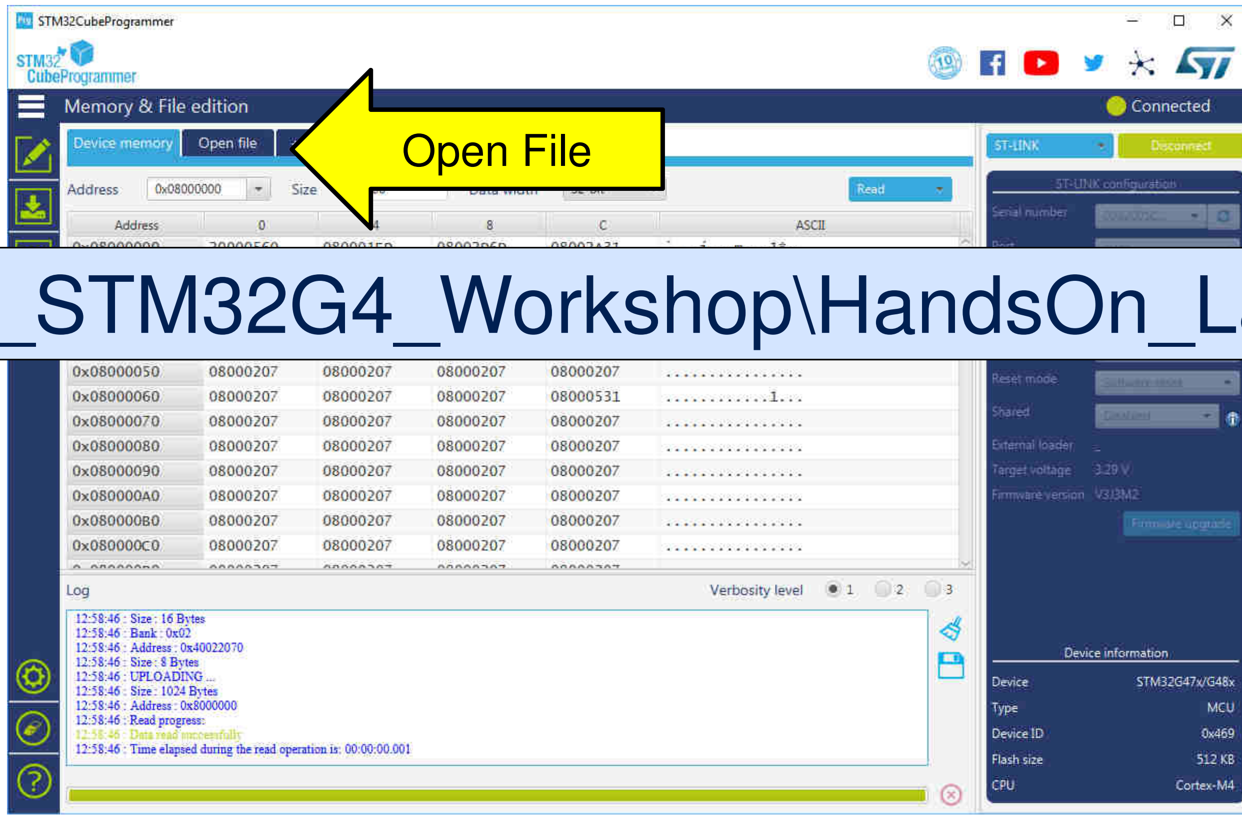
CubeProgrammer

Splash Page – Open File



CubeProgrammer

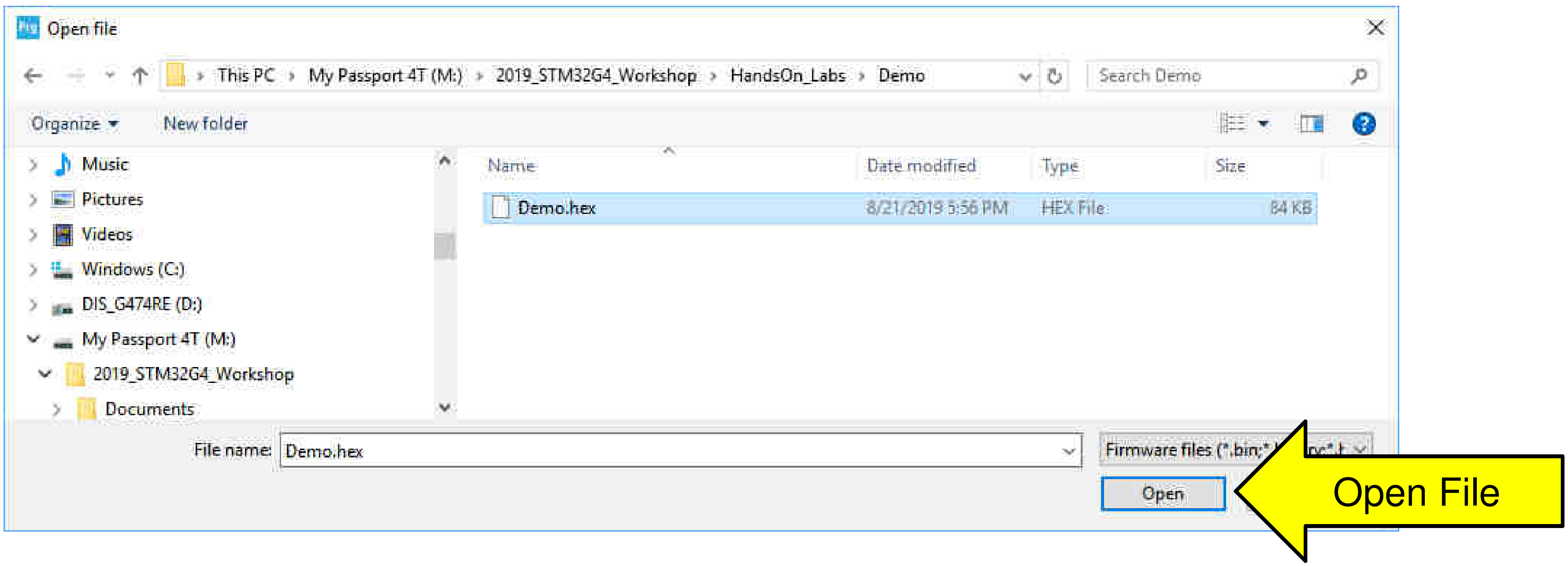
Find Demo.hex



2019_STM32G4_Workshop\HandsOn_Labs\Demo

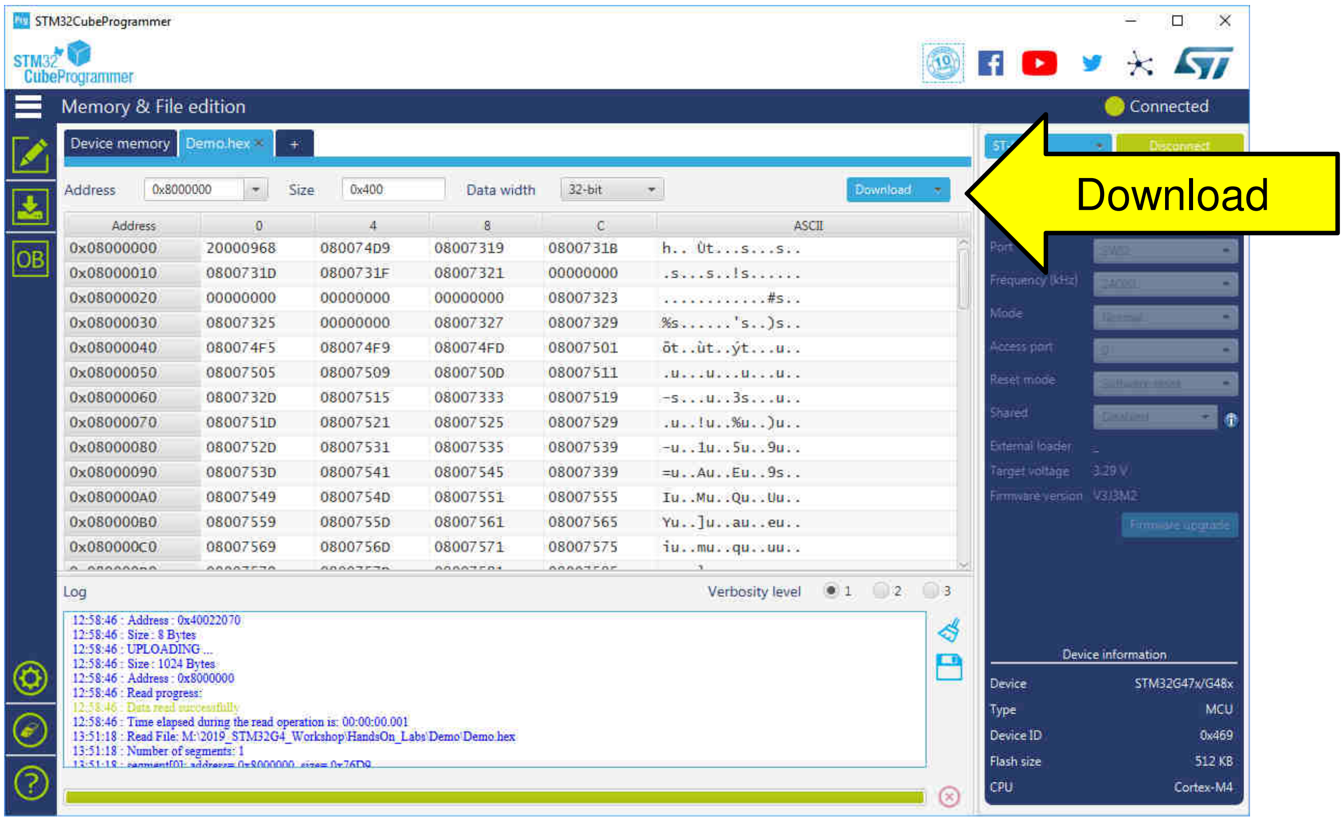
CubeProgrammer

File Folder



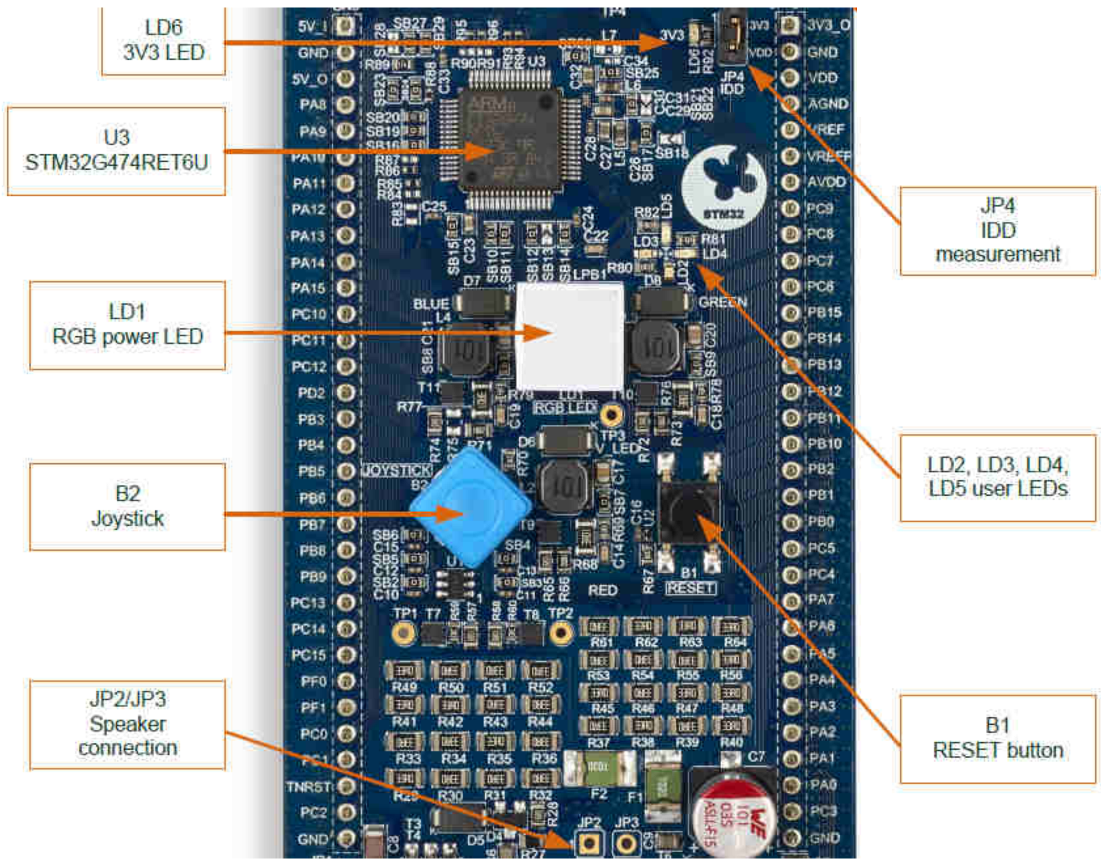
CubeProgrammer

Move Demo.hex from the PC to the Discovery board



B-G474E-DPOW1

Press RESET (Black Button) – LED’s cycle



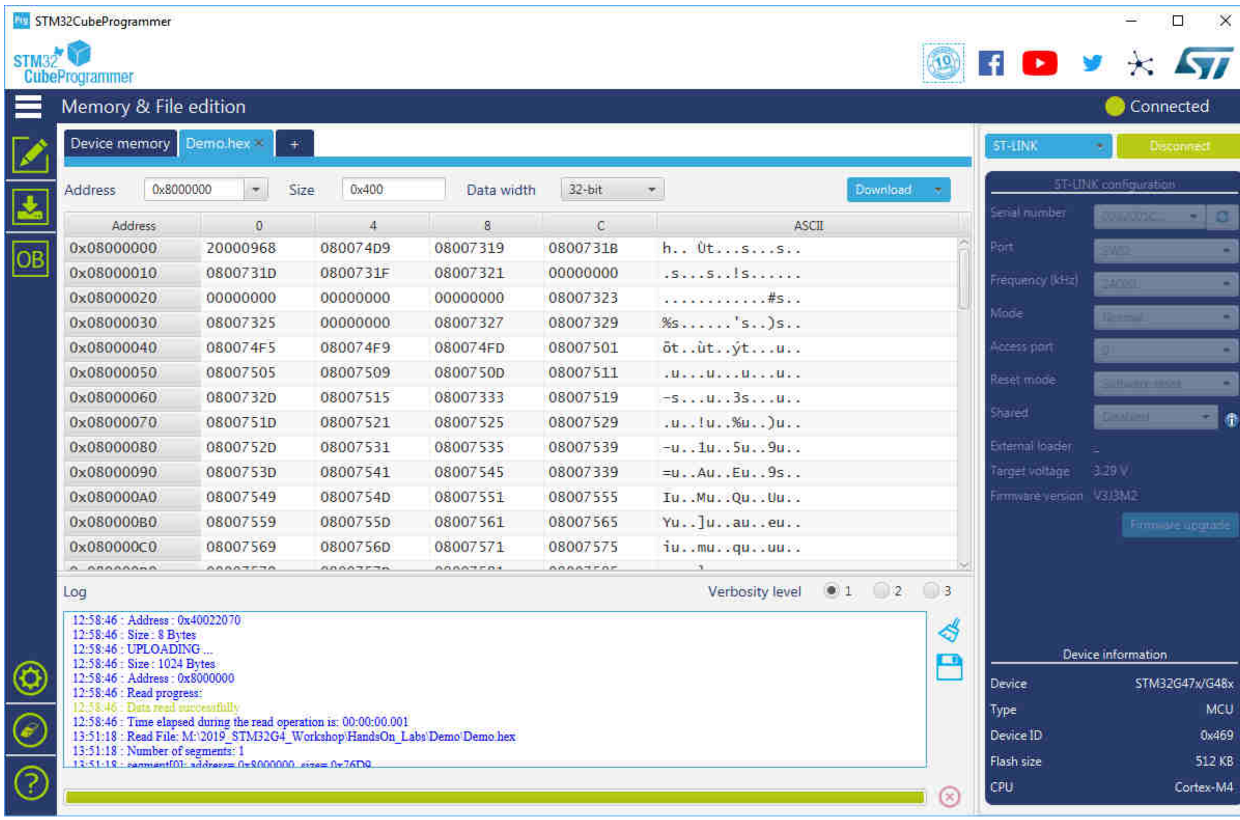
LED’s Cycle

PRESS Quickly

CubeProgrammer

Shut down the CubeProgrammer to release the VCOM port. …

Exit



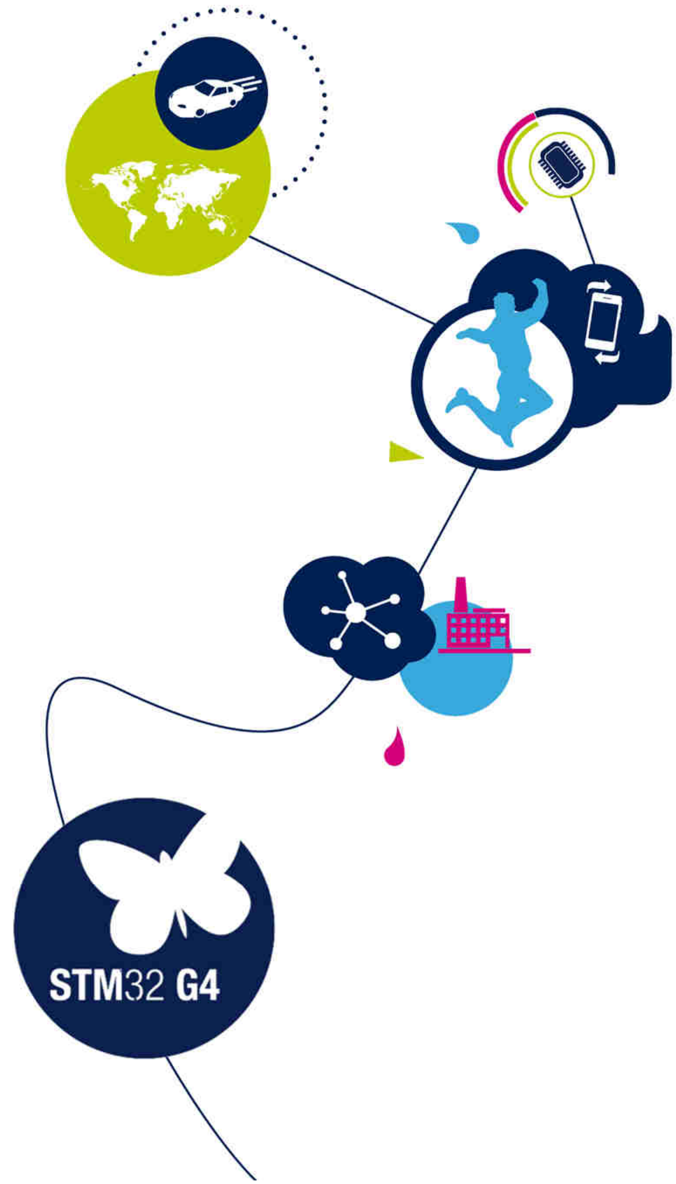
Getting back home – “CompletedLab”

For all of today’s Hands On Labs:

1. The *.hex file is available to you as a final product of the lab if you get lost along the way and still want to see the result of the lab.
2. The *.uvprojx file is available to you as a final result in case you want to pull it up in KEIL MDK-ARM and just walk through the files you see there.
3. The final STM32CubeMx *.ioc file for each lab is made available to you if you want to see the finished product of the lab or if you want to extend the lab with your own design ideas.

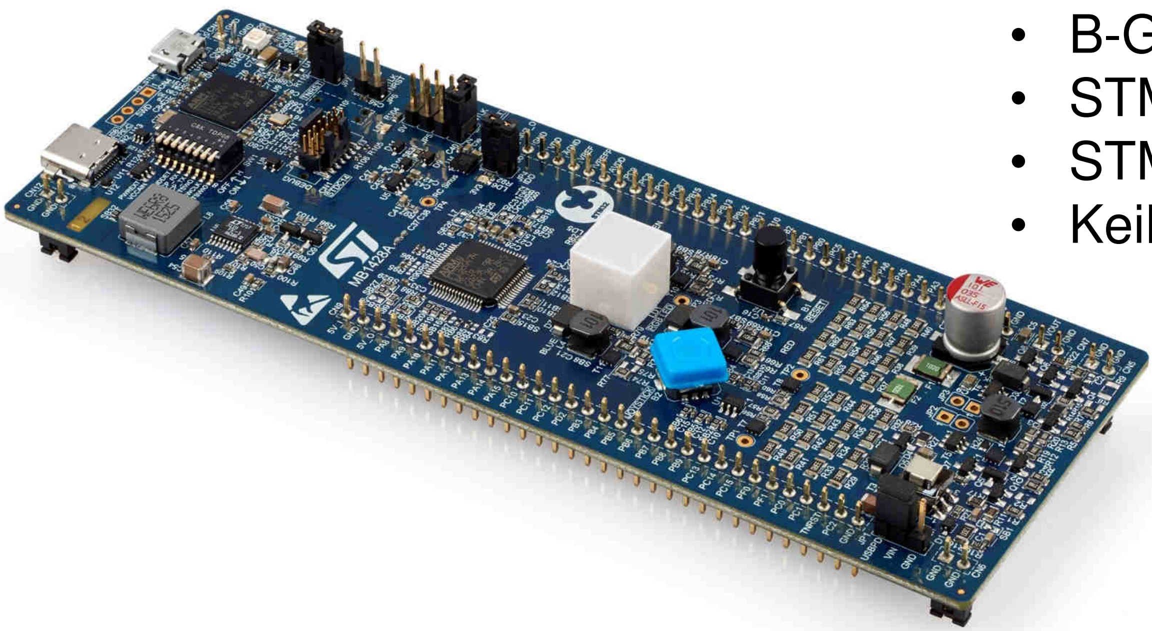
STM32G4 mixed-signal MCU workshop

Hands On:
ADC Gain Compensation



life.augmented

Systems Check



- USB cable
- B-G474E-DPOW1 Discovery Board
- STM32G4 Family Cube Library
- STM32CubeMx
- Keil MDK-ARM

Analog to Digital Converter

G4 ADC

- Up to 5 multi-input ADCs
- 4Msps - 12bit SAR
- 16bit hardware oversampling
- 2x to 256x oversample ratio
- DMA
- 3 Analog Watchdogs / ADC
- Single Ended or Differential inputs
- Independent clock

Internally connected to:

- Voltage reference
- Temperature Sensor
- 6 Op Amps
- Multiple Timers (+ HRTimer)

Analog to Digital Converter

New Features – Gain and Offset Compensation

Gain Compensation:

- Activated on all converted data
- Effective for oversampling
    - Calculation is done after accumulation
    - And after right-shifting

The lab will use this feature to show the voltage directly in millivolts without need for post computing of the converted data

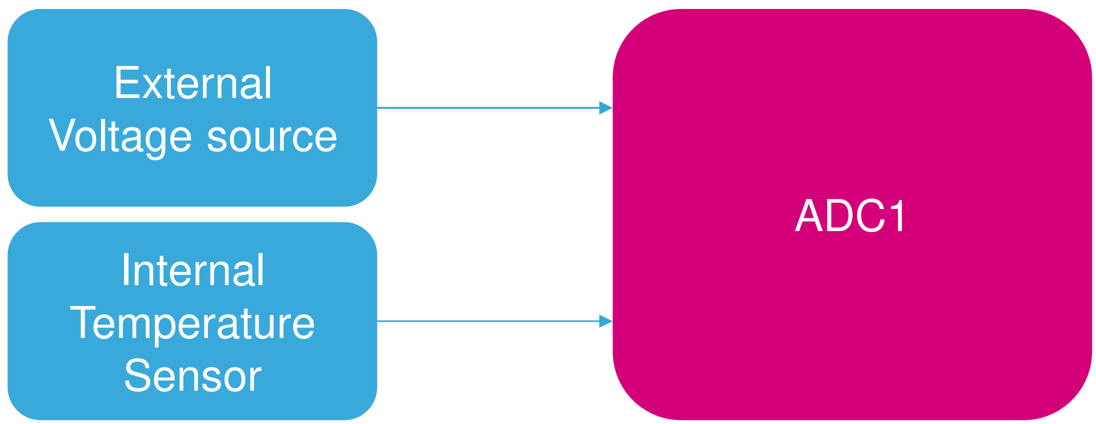
Offset Compensation:

- Analog Watchdog performed on unsigned values after gain and offset compensation

The lab directly translates the conversion data from the ADC range to the application range

Analog to Digital Converter

Block Diagram



ADC Gain Compensation

- Setting the GCOMP bit in ADC_CFGR2 activates the gain compensation
- The compensation is applied to all the converted data.
- After each conversion, data is calculated using the formula:

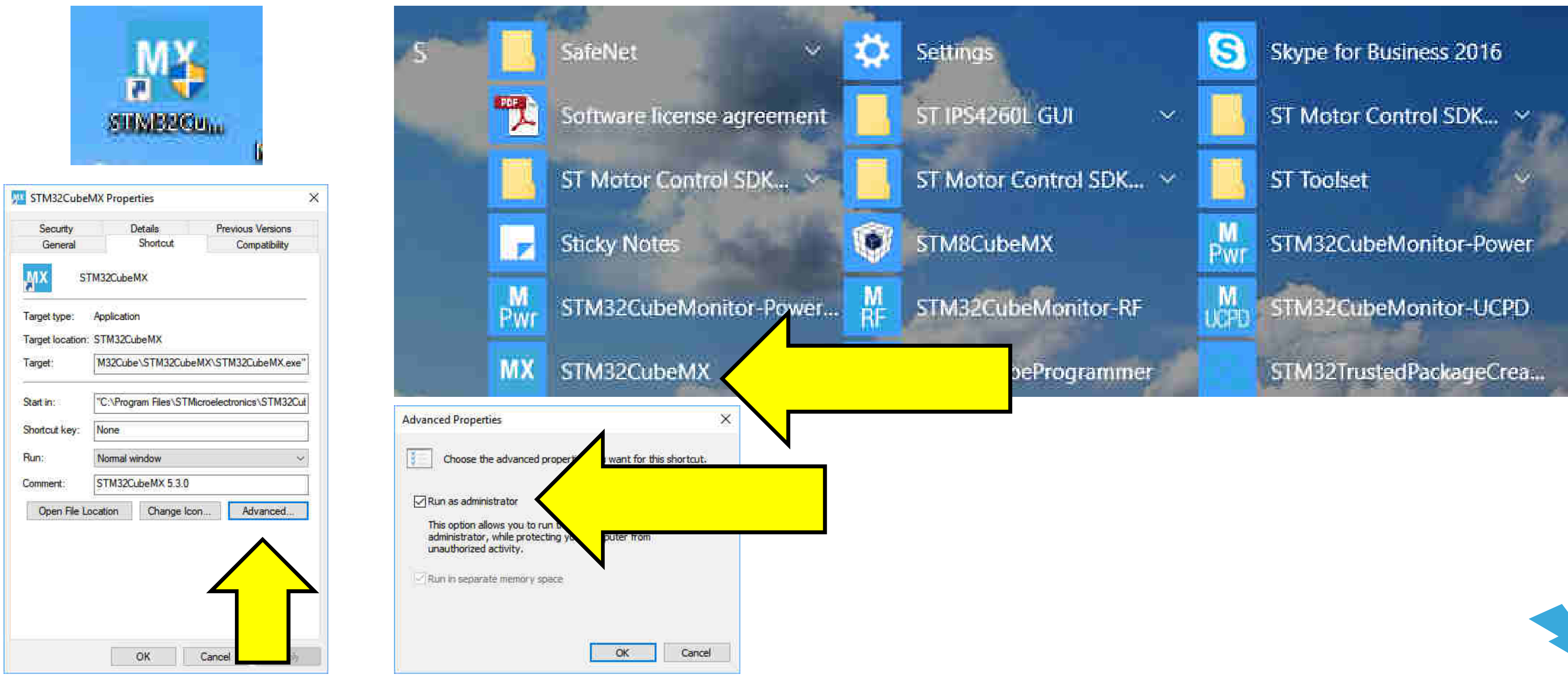
$$Data = Data(ADC_{Result})x\,\frac{(GCOMPCoeffcient)}{4096}$$

Where the GCOMPCoeffcient is a 14 bit value (16384).
Since this value is divided by 4096, the gain factor ranges from 0 to 3.999756

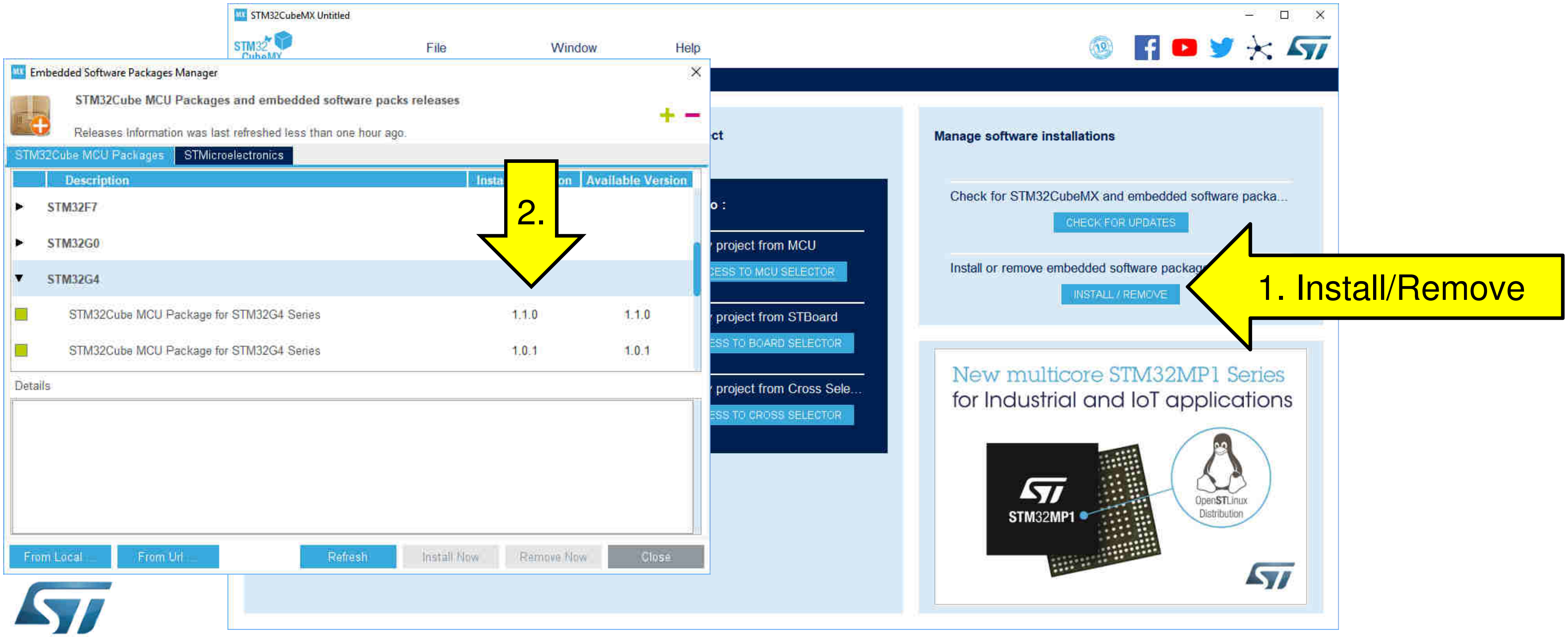
STM32 G4

STM32CubeMx

Start up



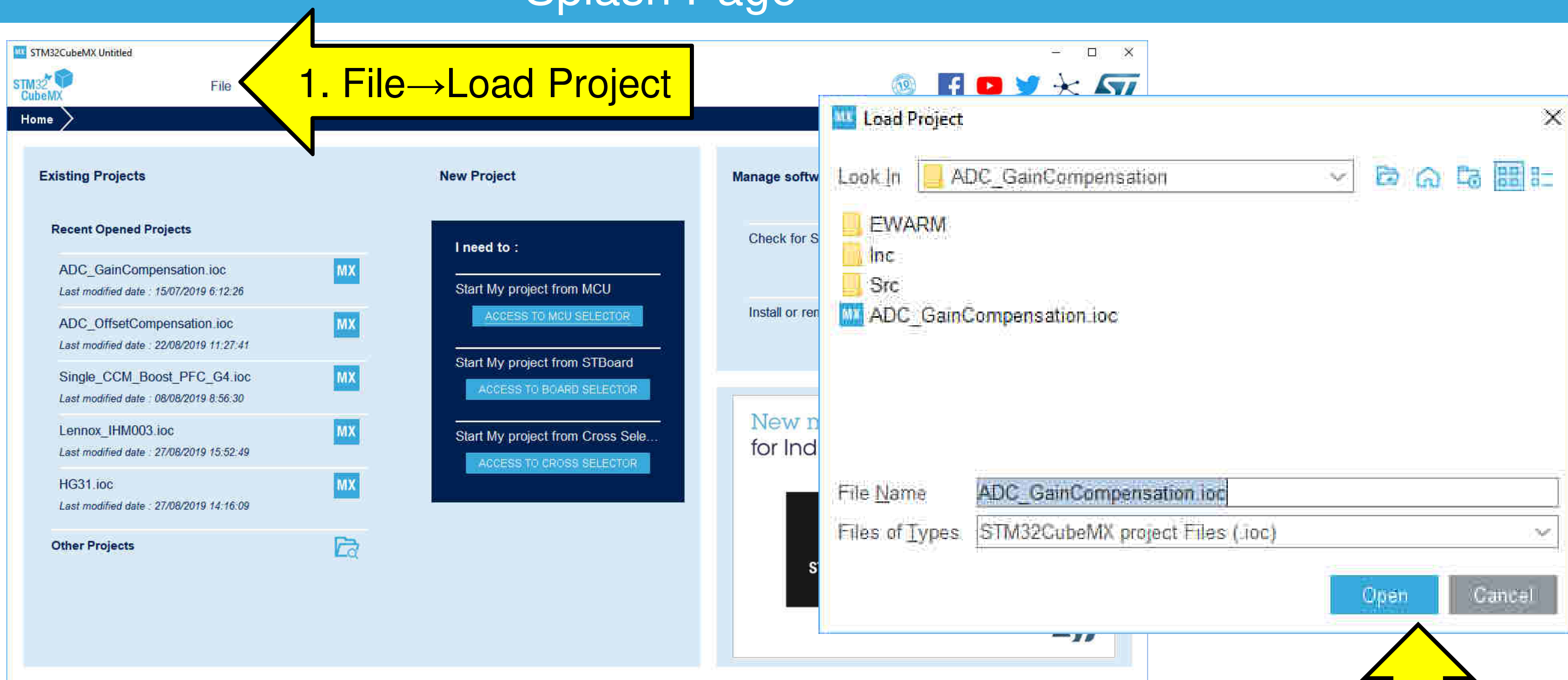
Splash Page – G4 Library Installed?



life.augmented

STM32CubeMx

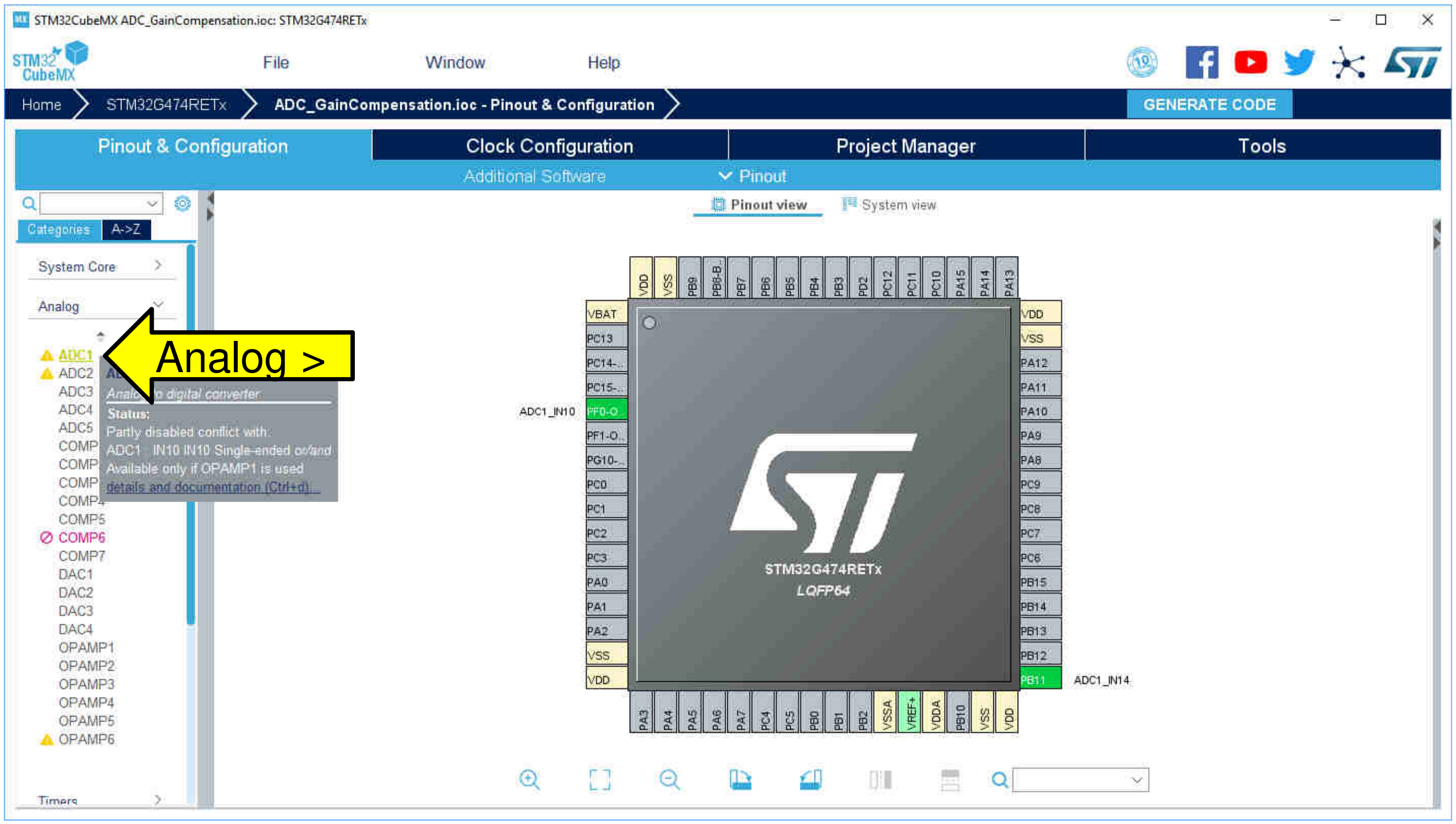
Splash Page



C:\...\STM32Cube\Repository\STM32Cube_FW_G4_V1.1.0\Projects \B-G474E-DPOW1\Examples\ADC\ADC_GainCompensation

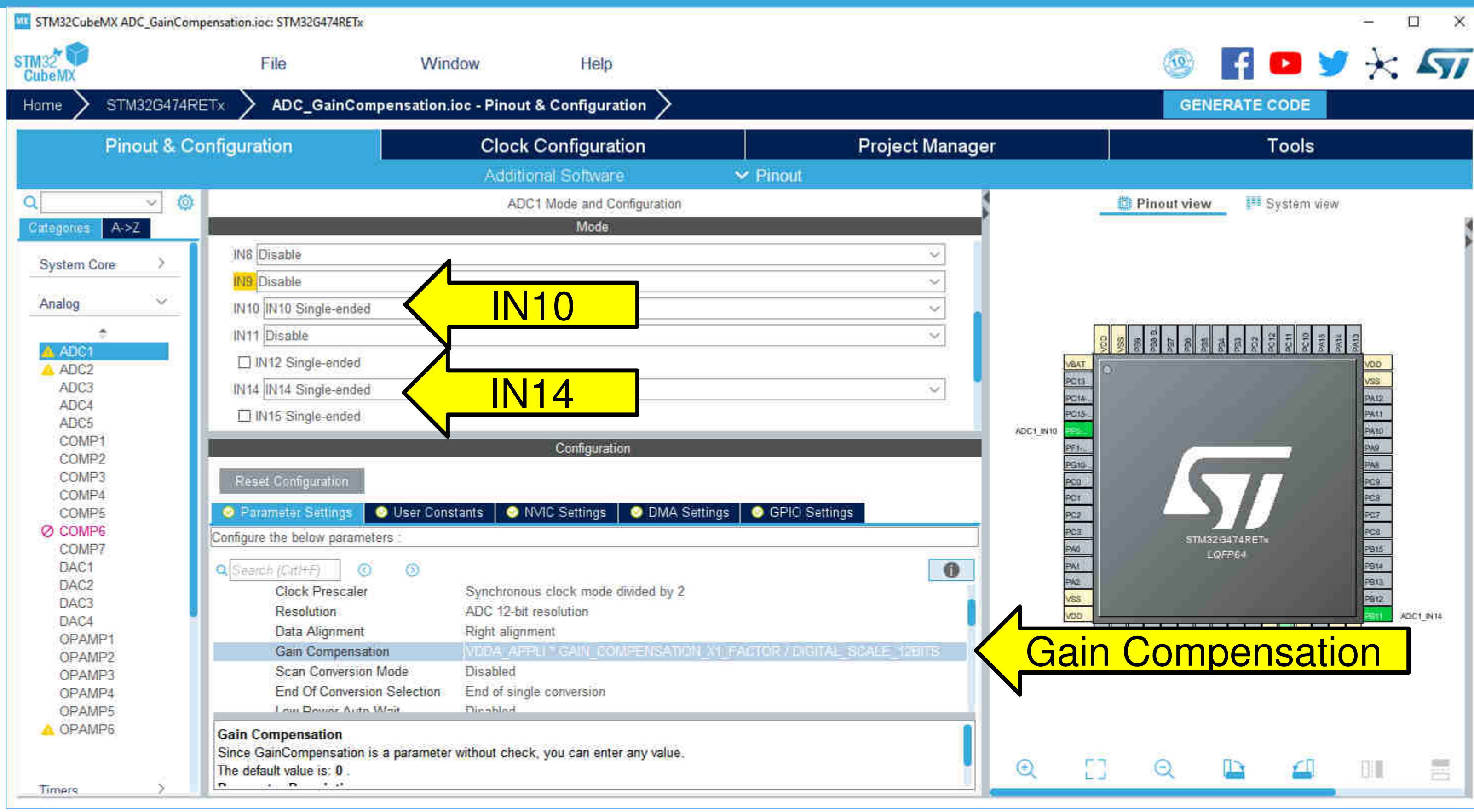
STM32CubeMx

Graphic Design Representation



STM32CubeMx

ADC1_IN10, ADC1_IN14, Gain Compensation

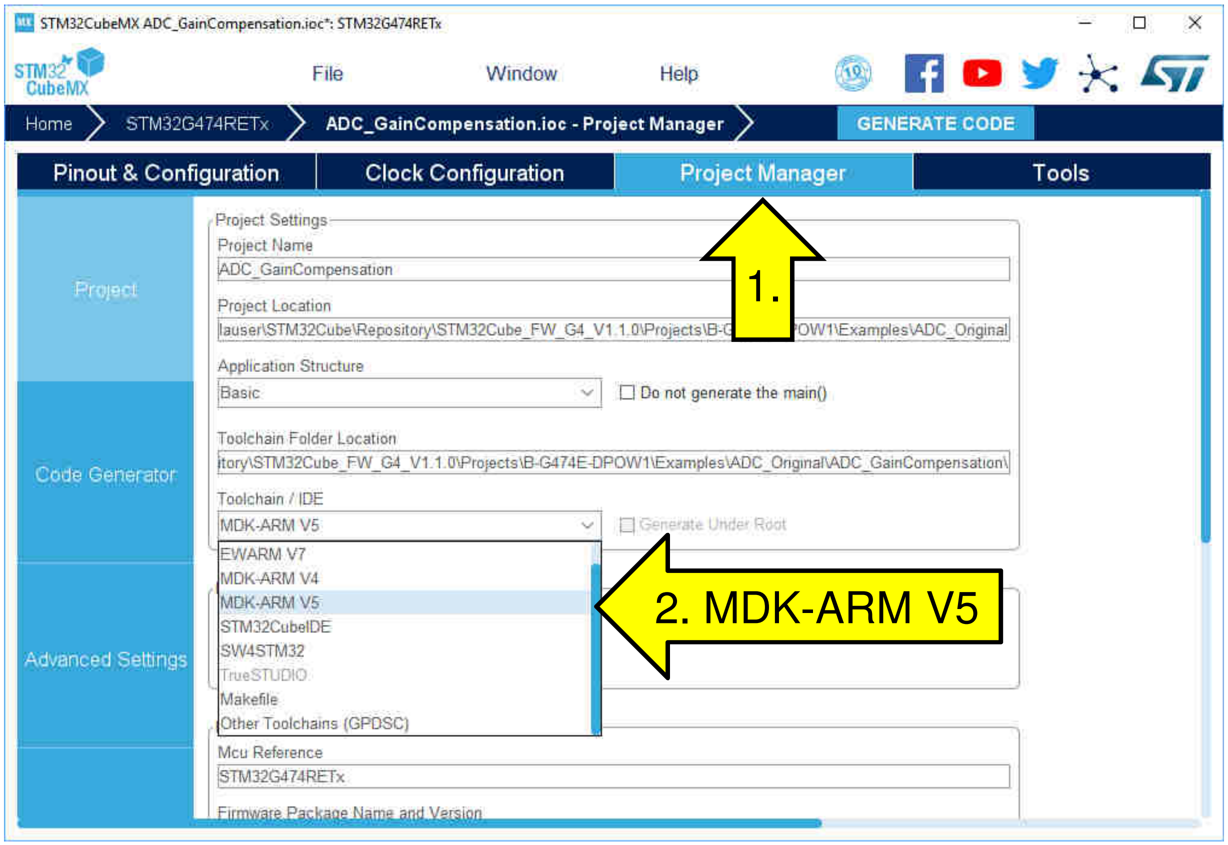


Gain Compensation Expanded

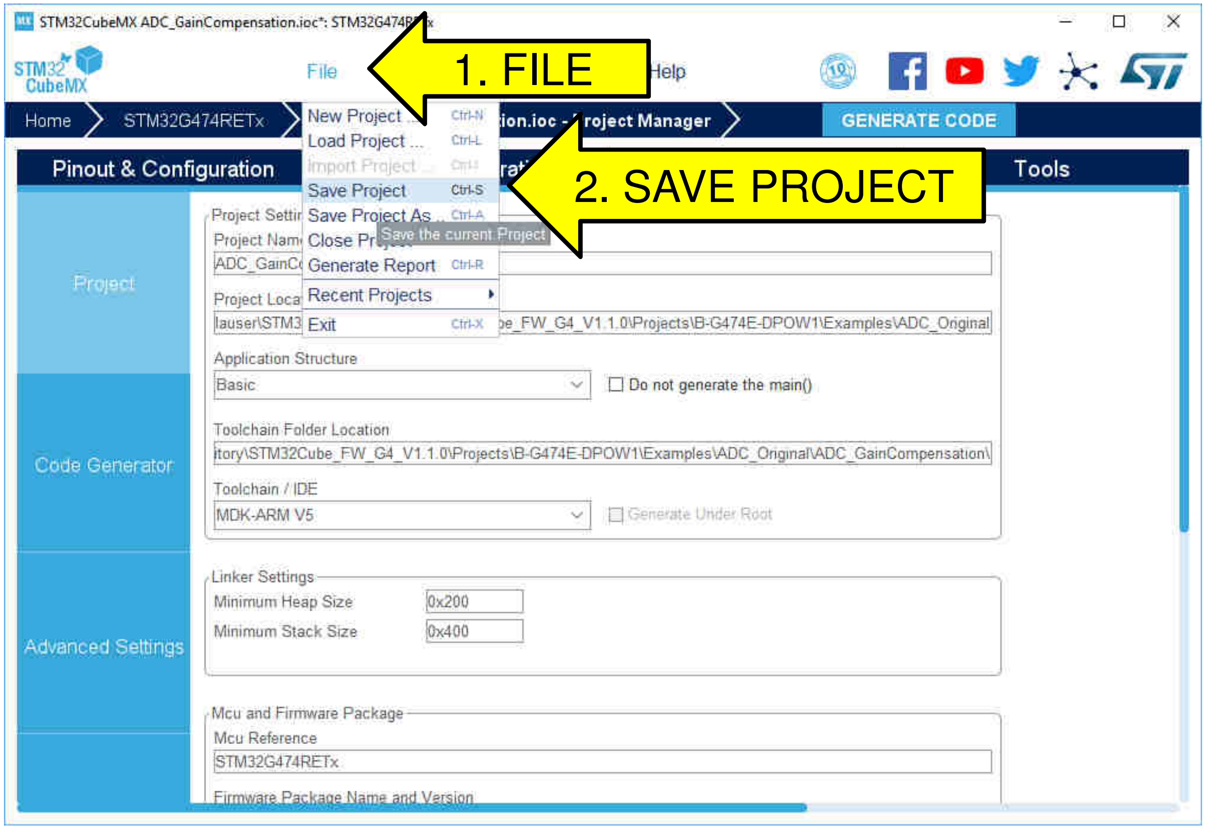
Data Alignment	Right alignment
Gain Compensation	VDDA_APPLI * GAIN_COMPENSATION_X1_FACTOR / DIGITAL_SCALE_12BITS
Scan Conversion Mode	Disabled

$$Data = Data(ADC_{Result})x\frac{(GCOMPCoeffcient)}{4096}$$

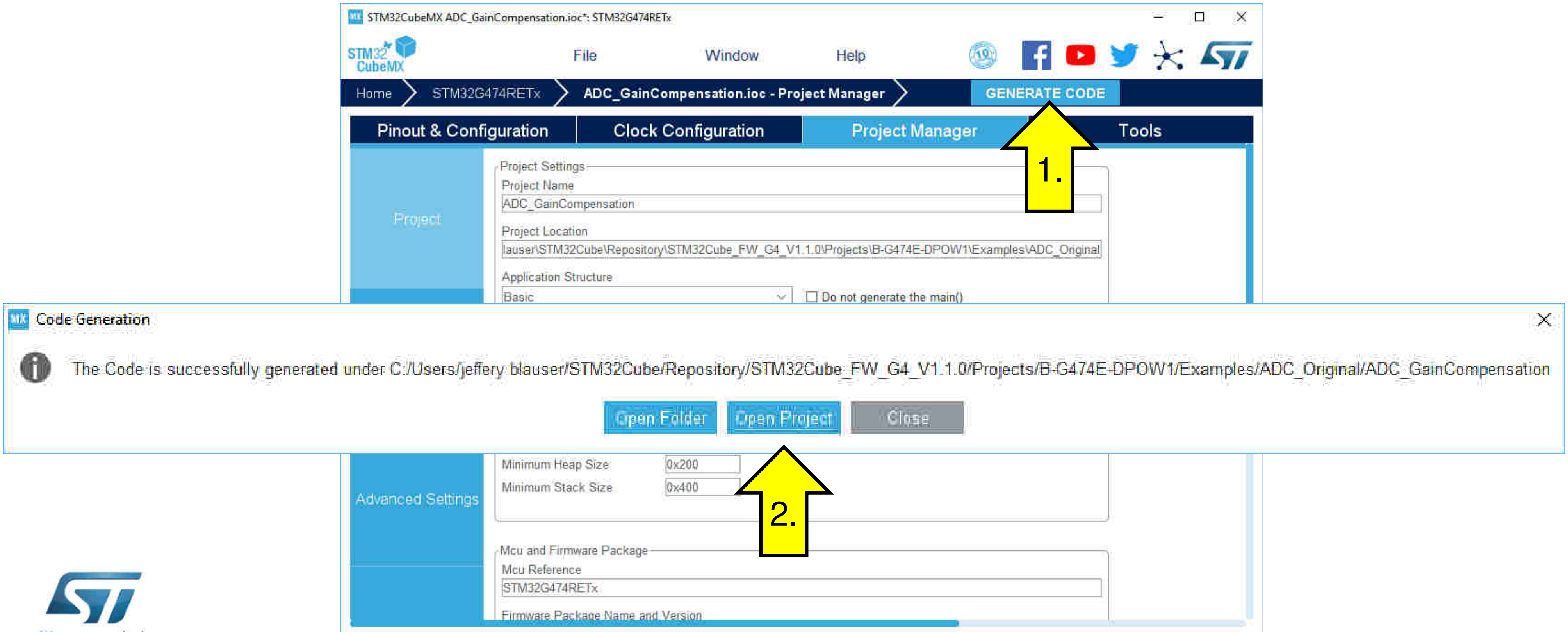
Project Manager – Keil MDK-ARM V5



FILE→SAVE PROJECT



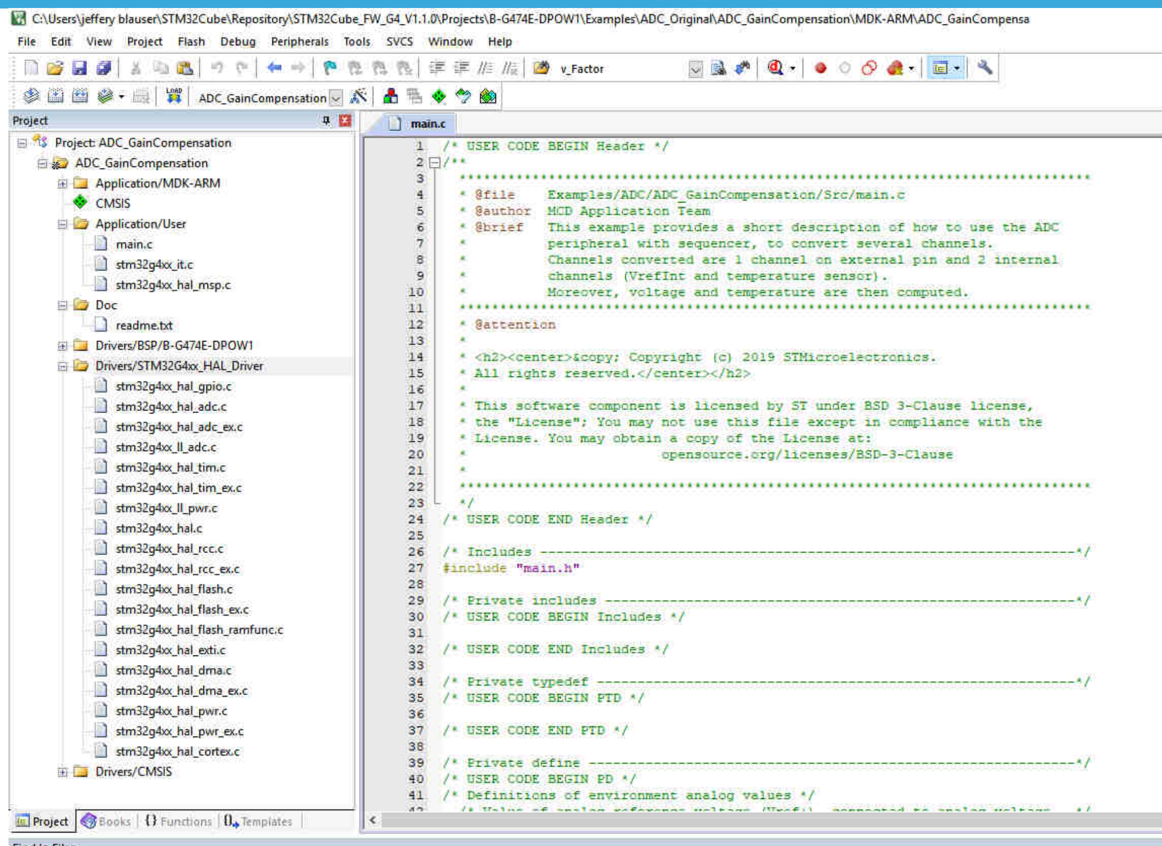
Generate Code



life.augmented

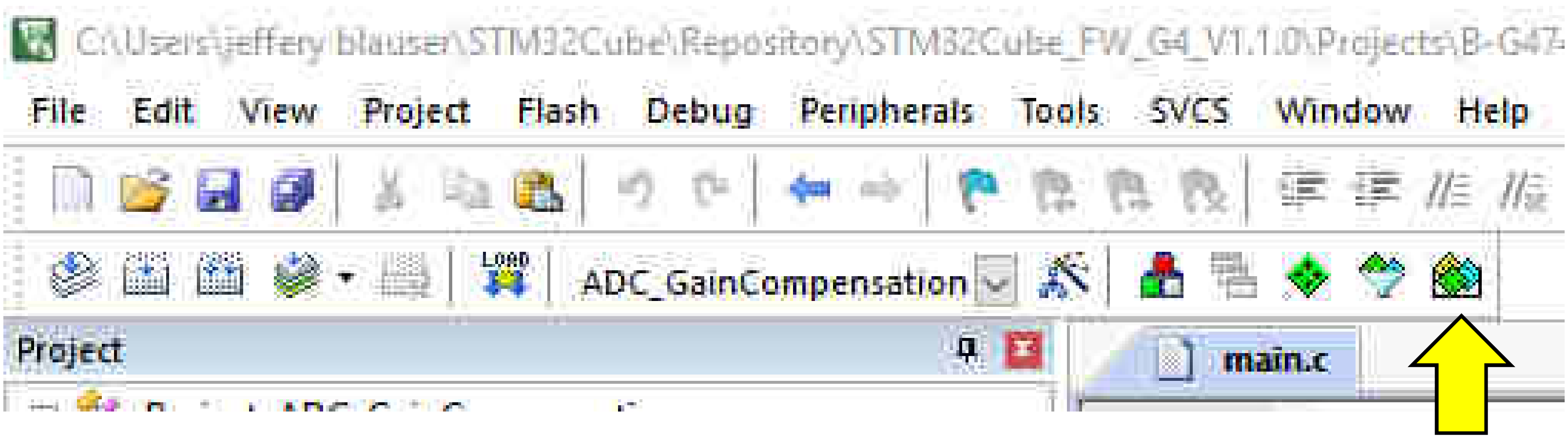
Keil MDK-ARM

Keil Start up page

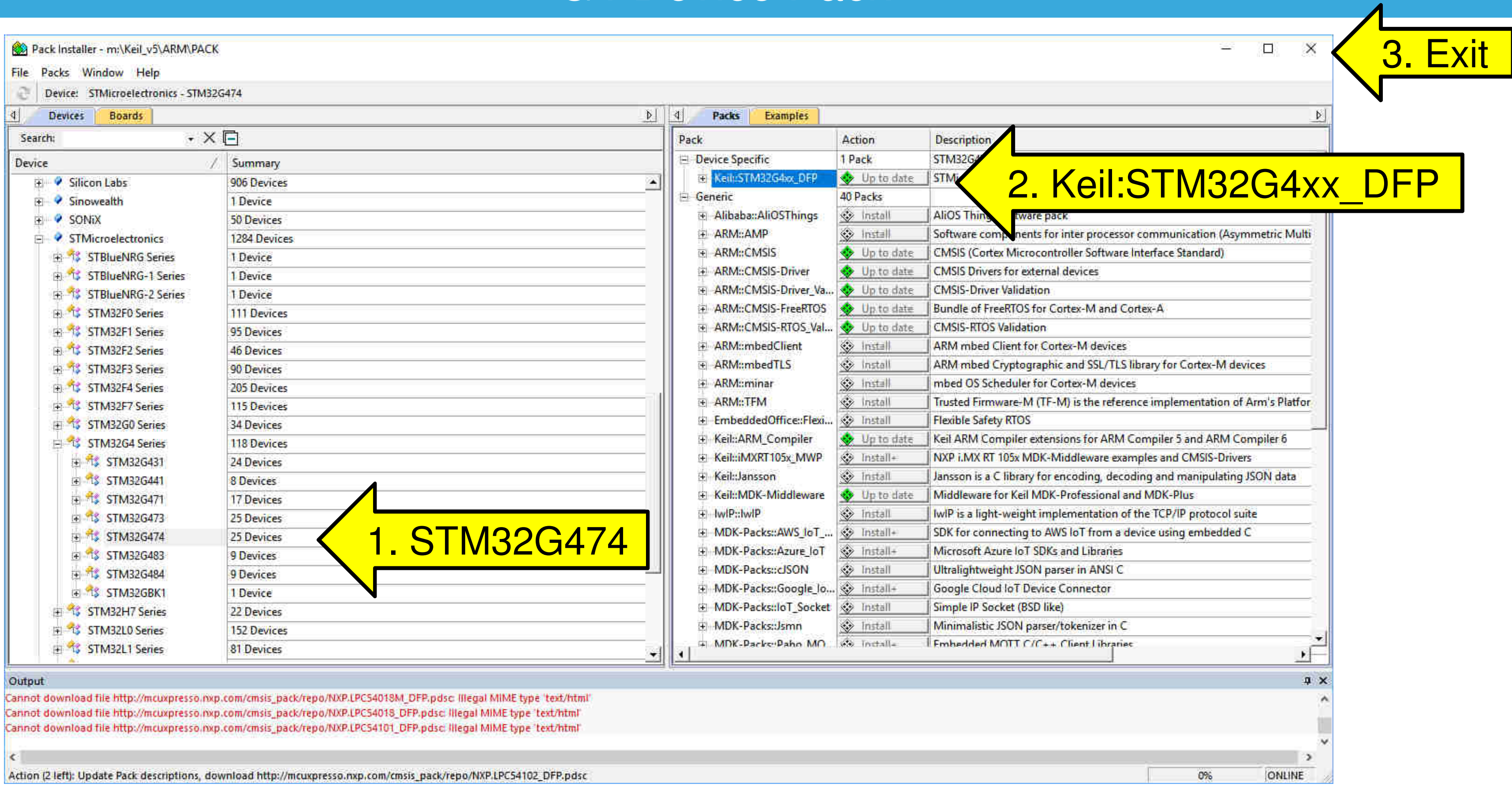


Keil MDK-ARM

Check the Device Pack

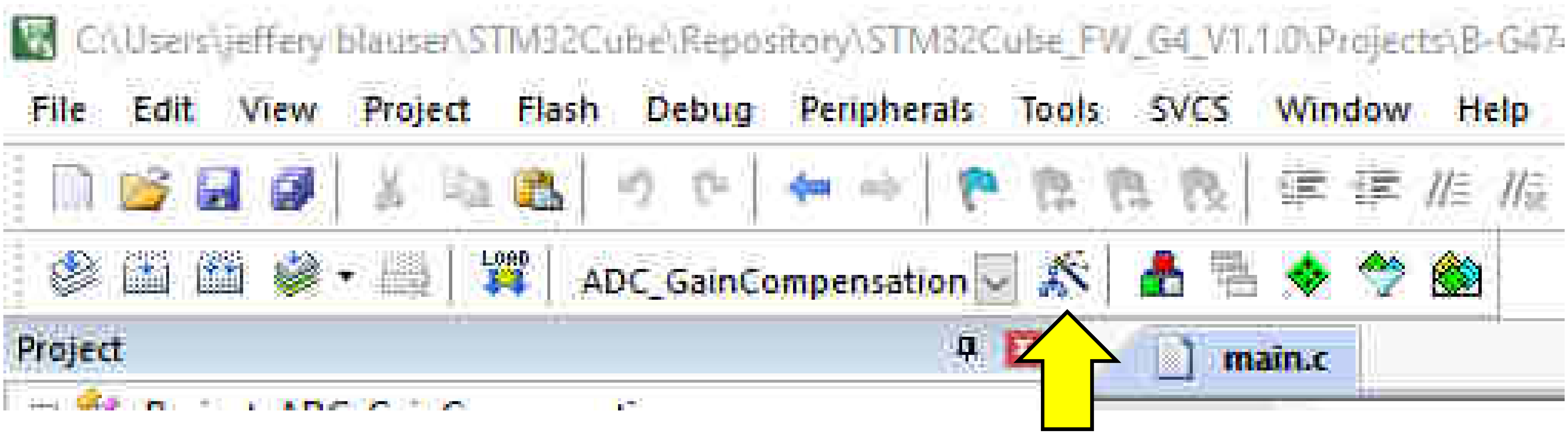


G4 Device Pack

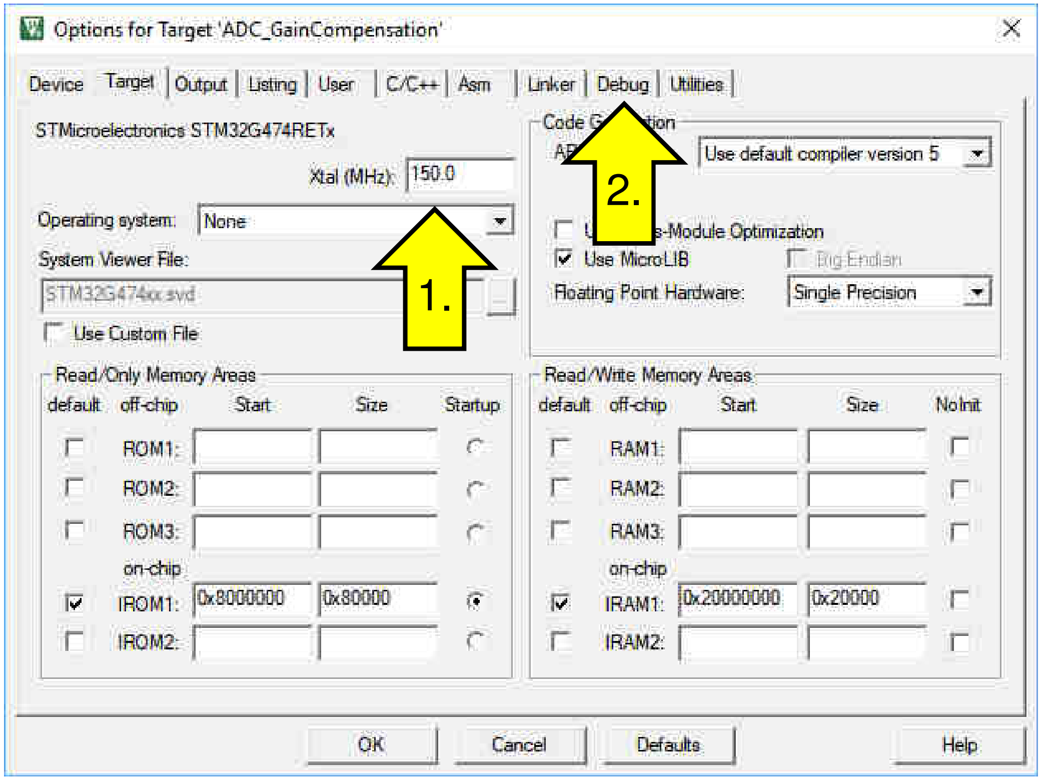


Keil MDK-ARM

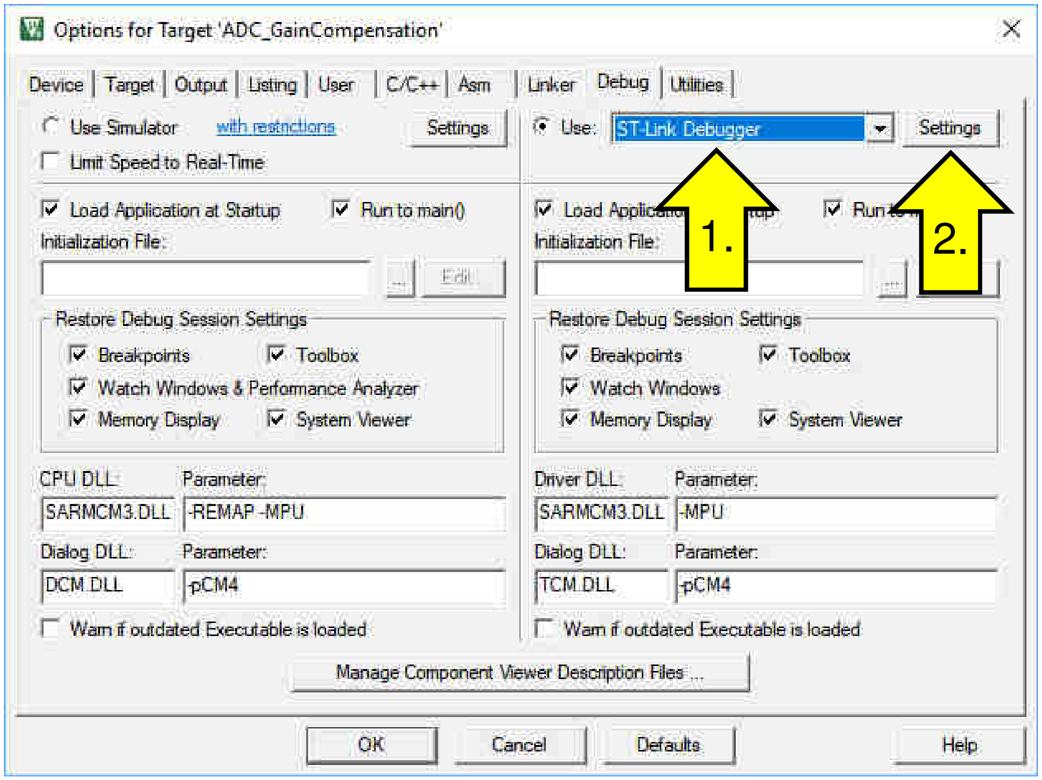
Check the Options



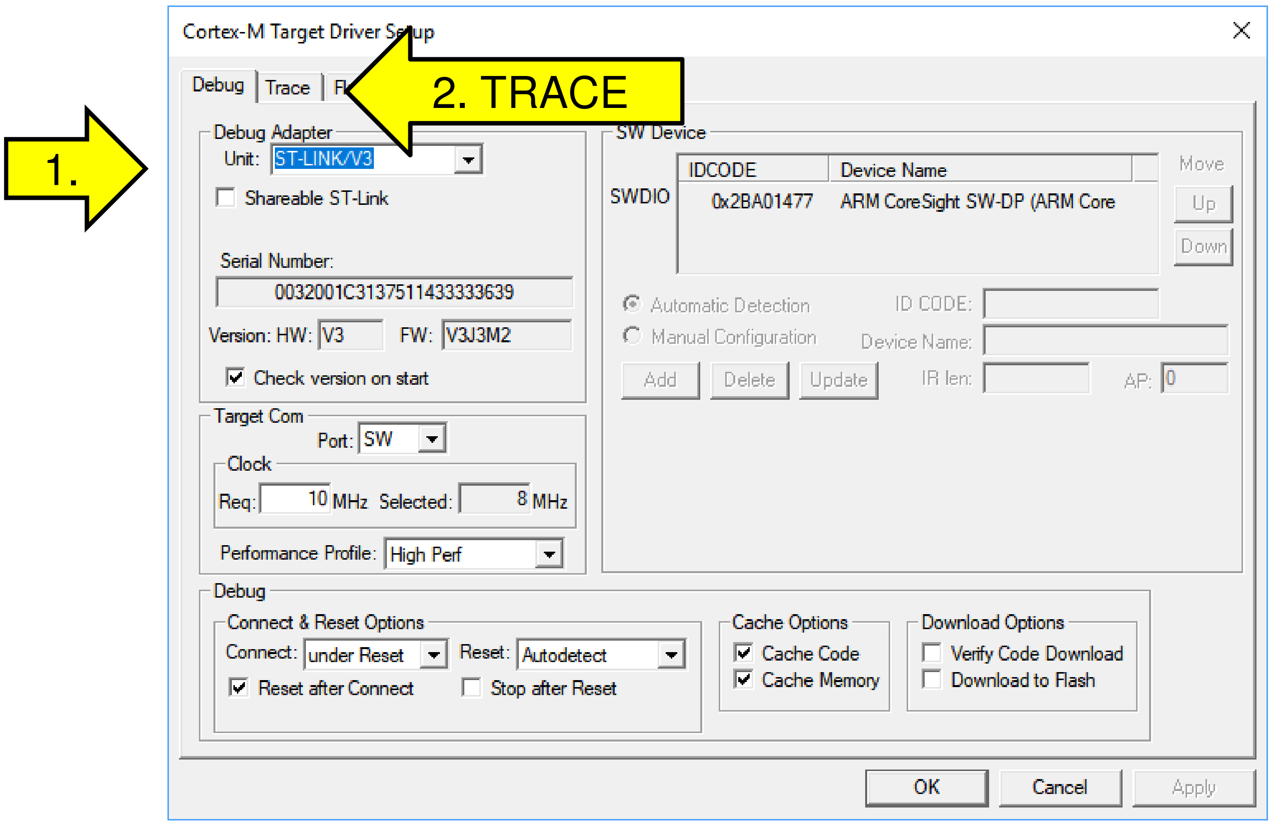
Clock speed and Debug



ST-Link Debugger - Settings

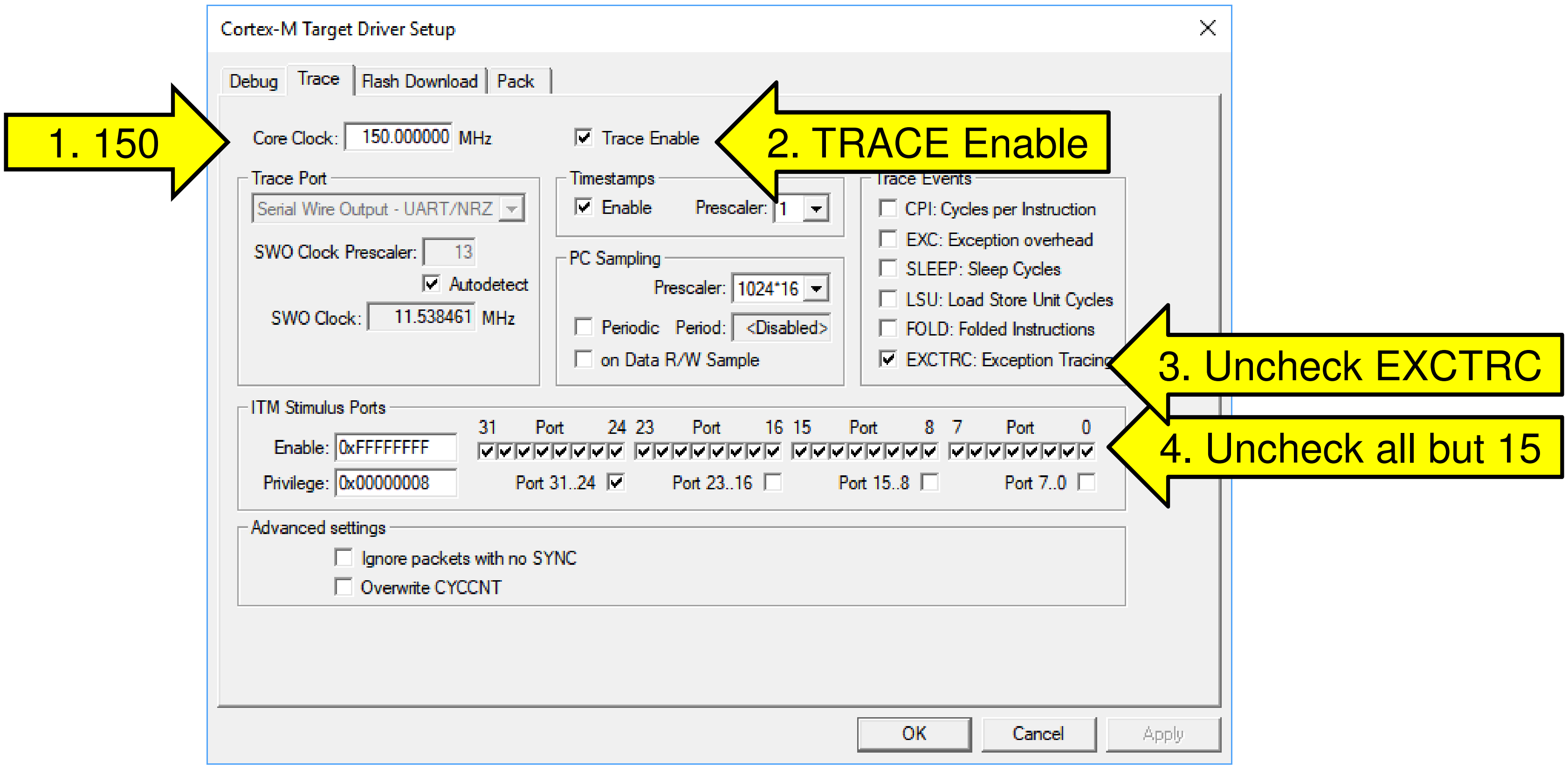


ST-Link/V3 & TRACE



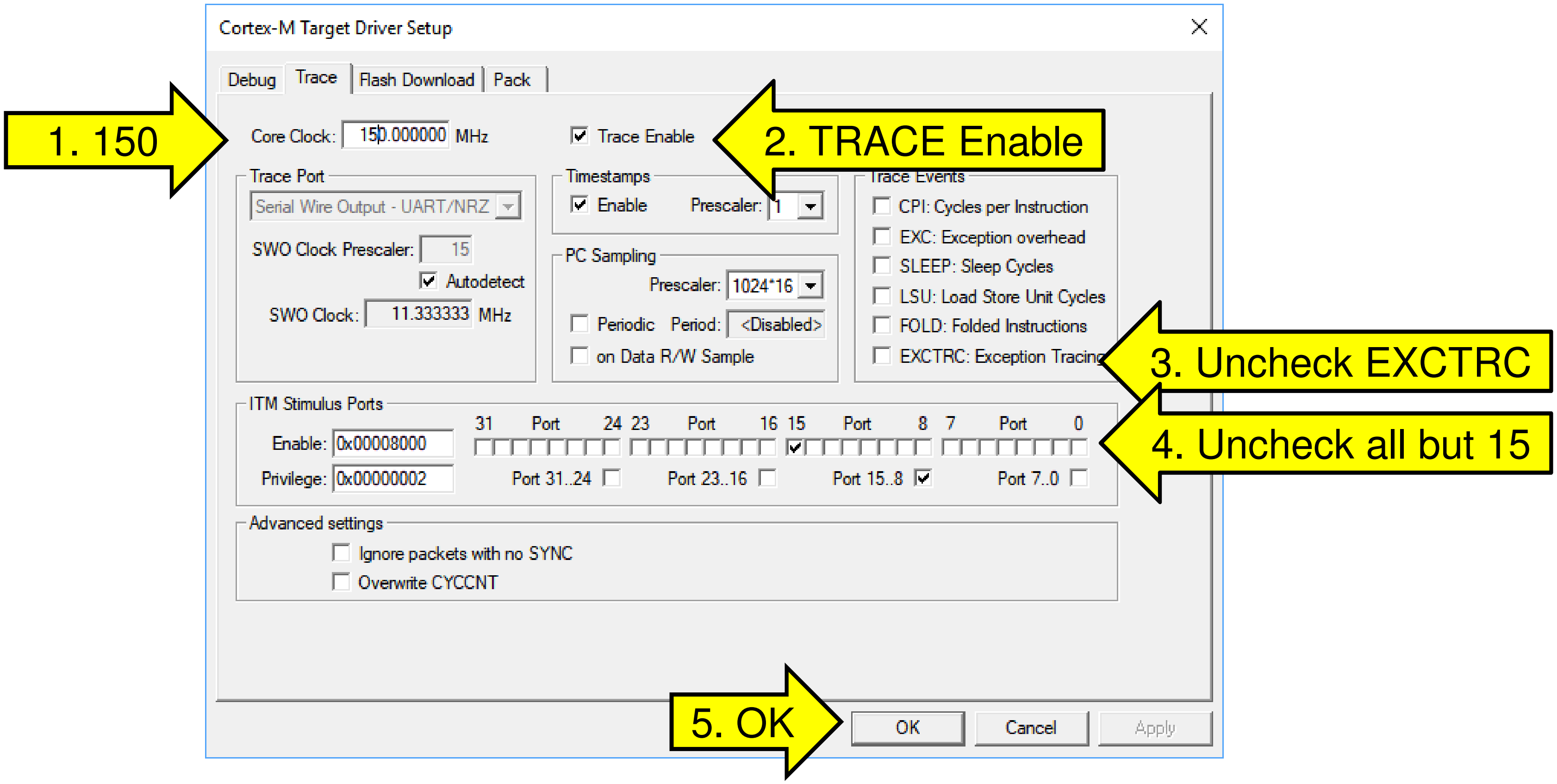
Keil MDK-ARM

Core Clock & TRACE Enable - Before



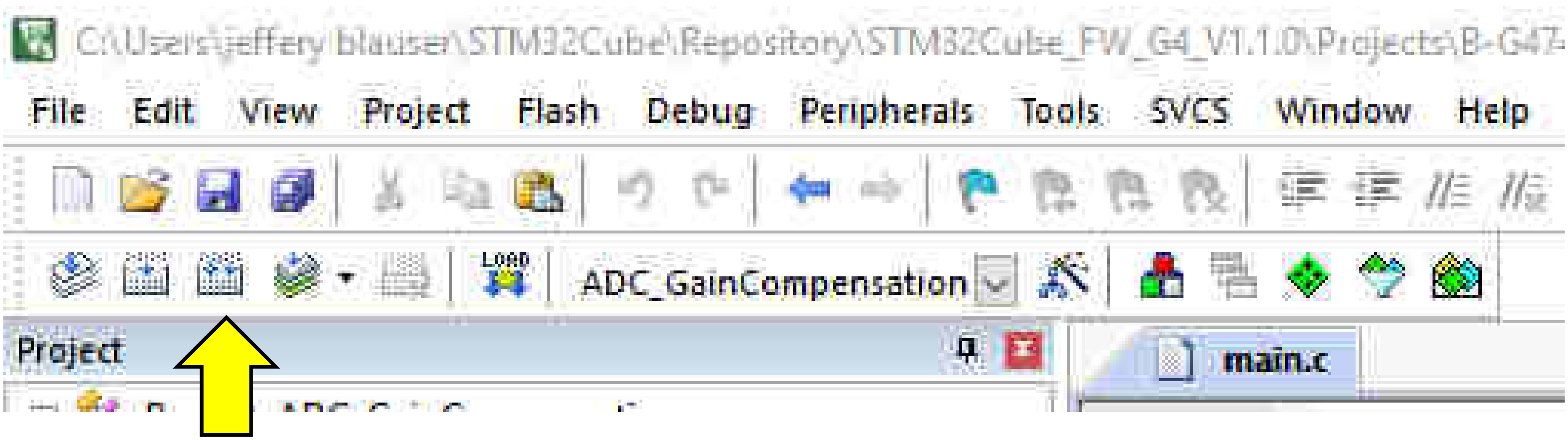
Keil MDK-ARM

Core Clock & TRACE Enable - After

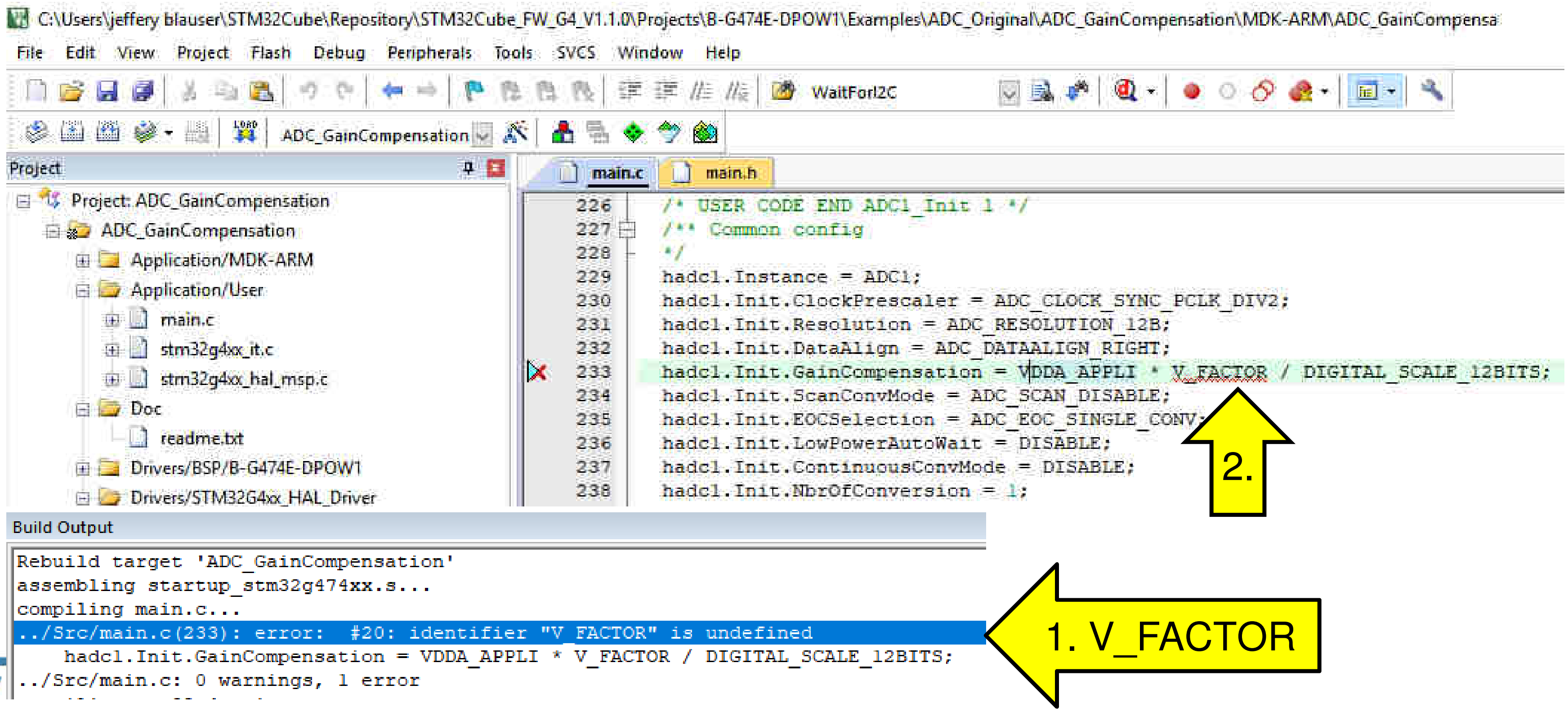


Keil MDK-ARM

Build

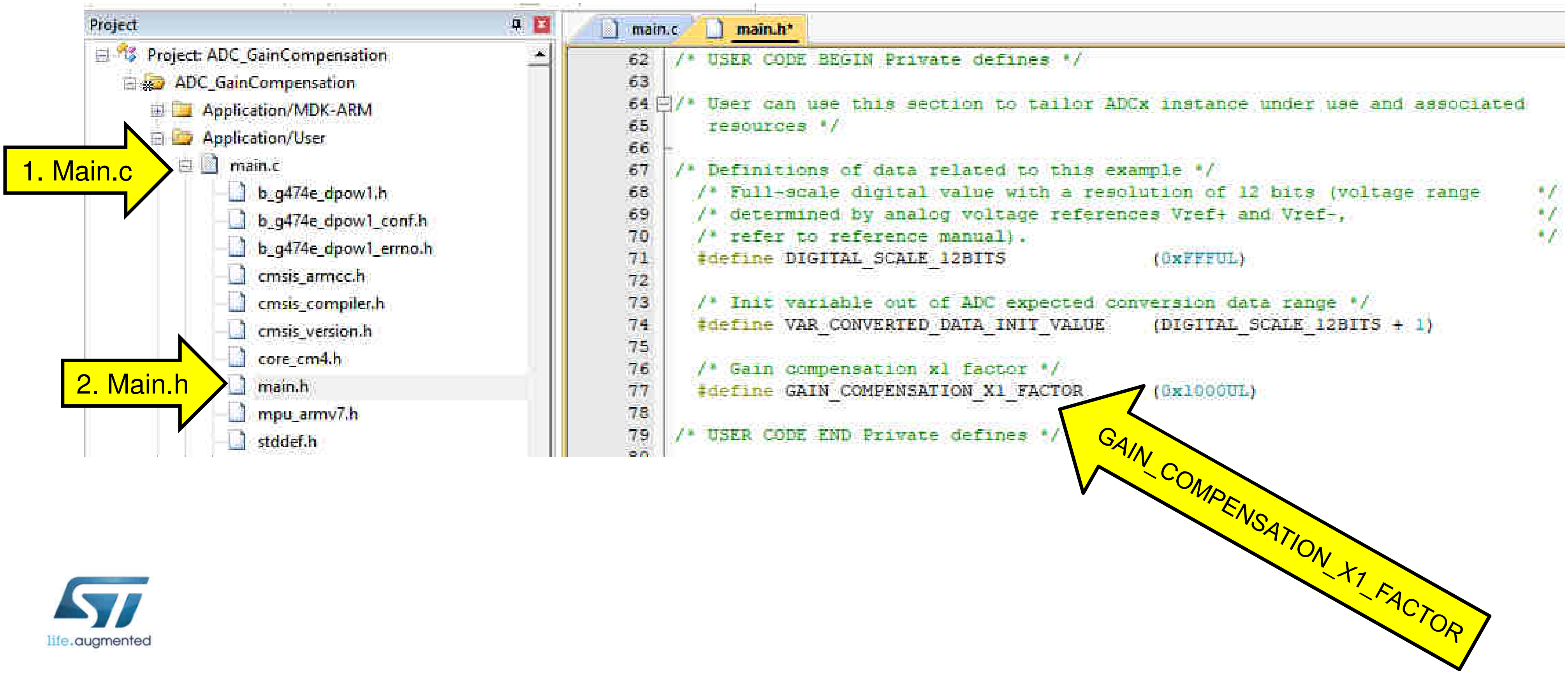


Error about line 233 in main.c



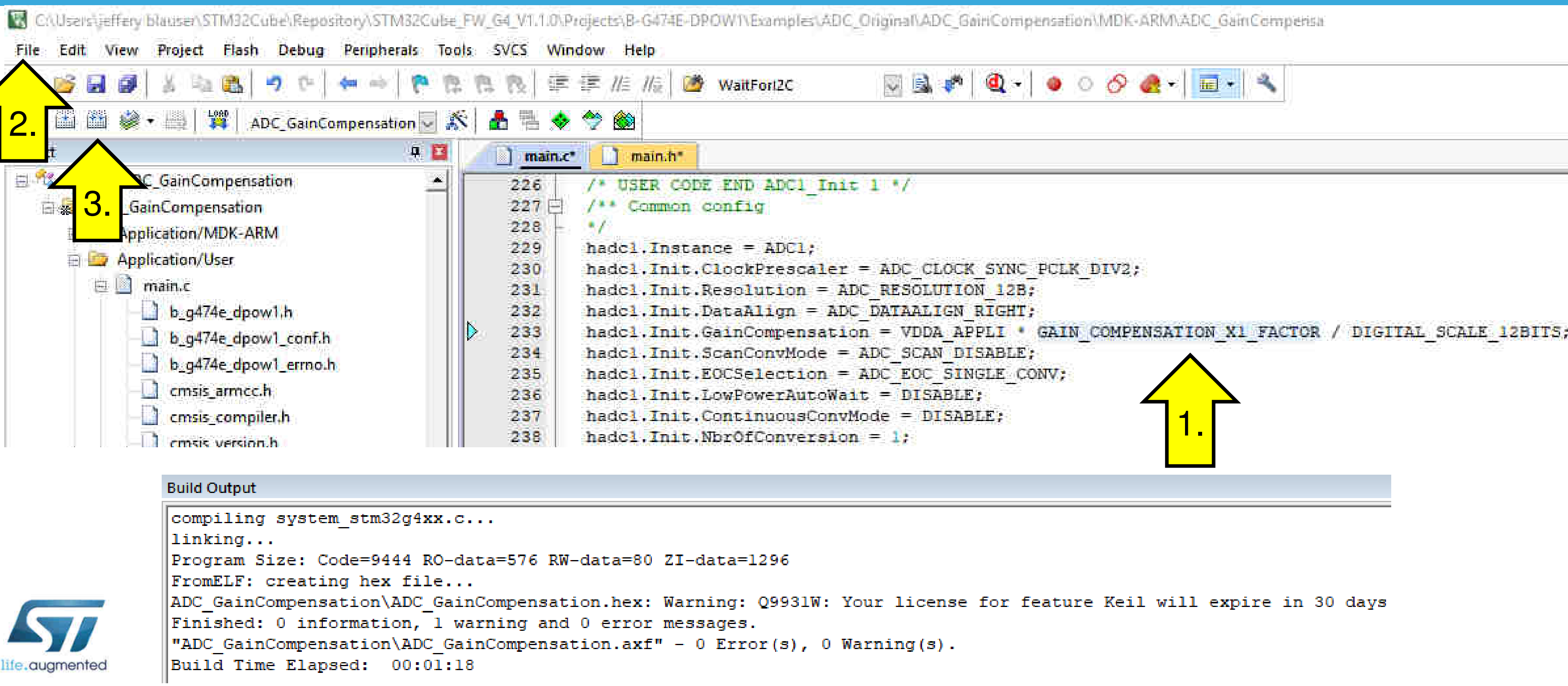
Keil MDK-ARM

About line 77 in main.h – Copy - GAIN_COMPENSATION_X1_FACTOR



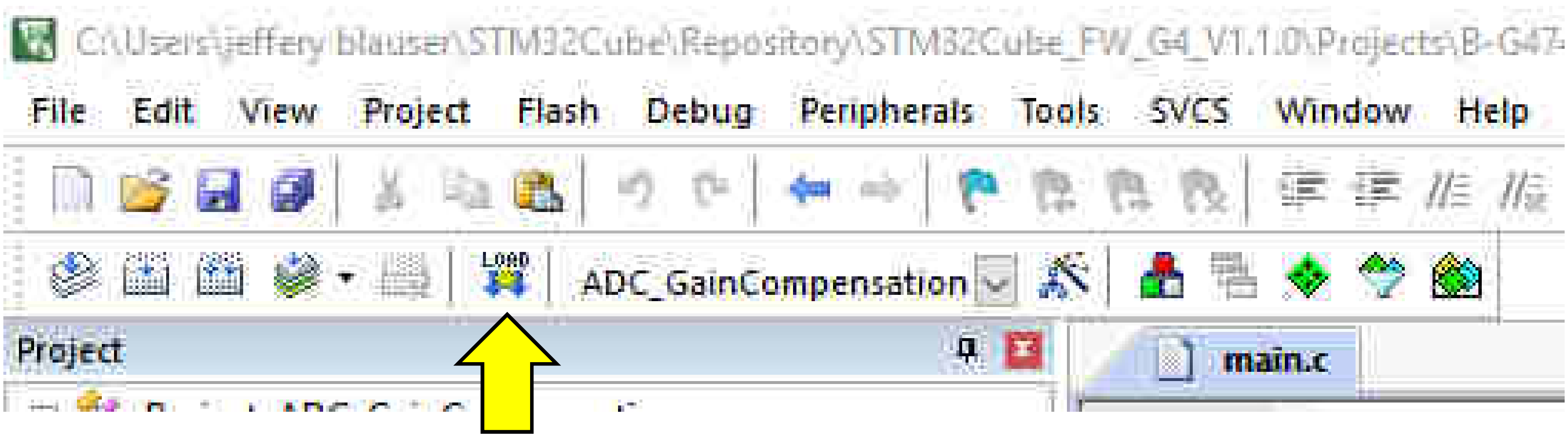
Keil MDK-ARM

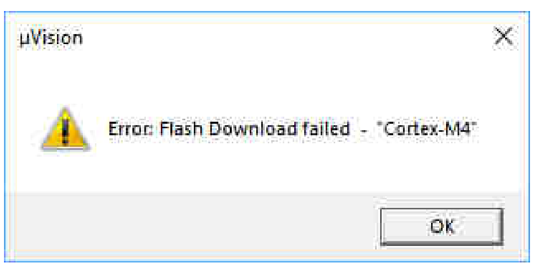
Paste into line 233 in main.c, FILE→SAVE, BUILD

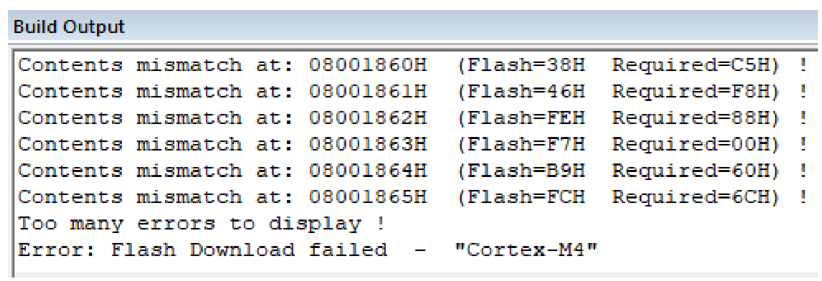


Keil MDK-ARM

Load

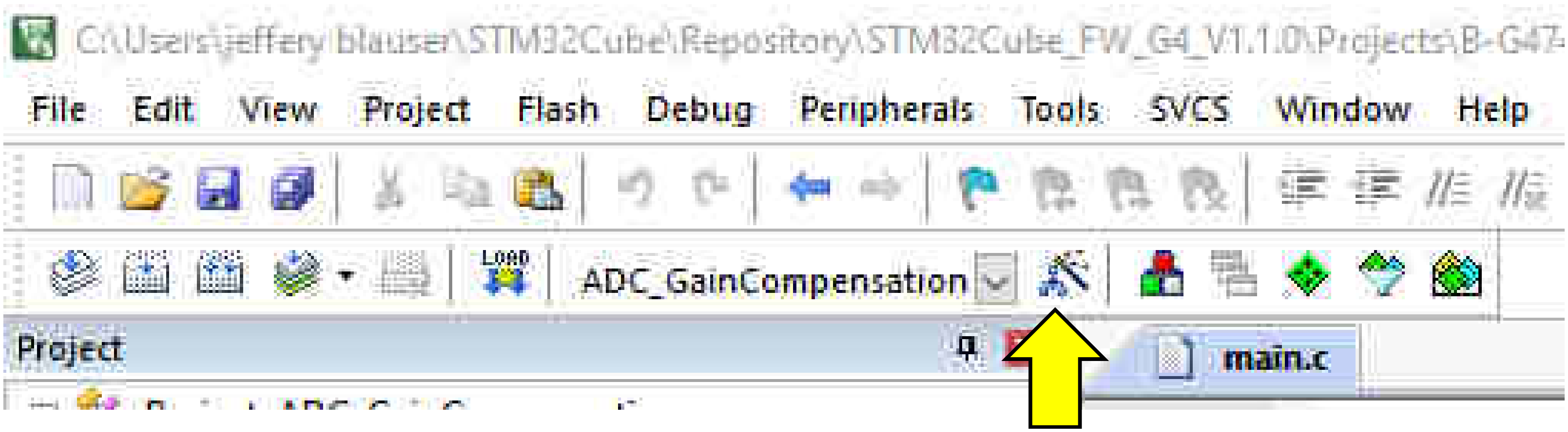




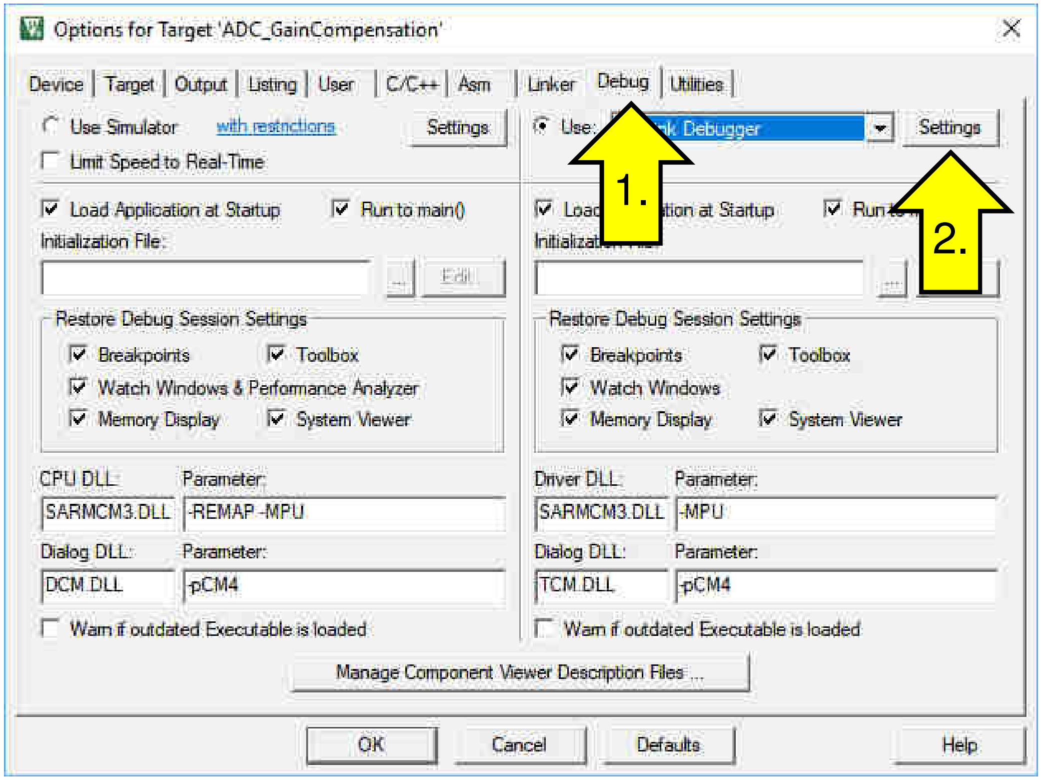


Keil MDK-ARM

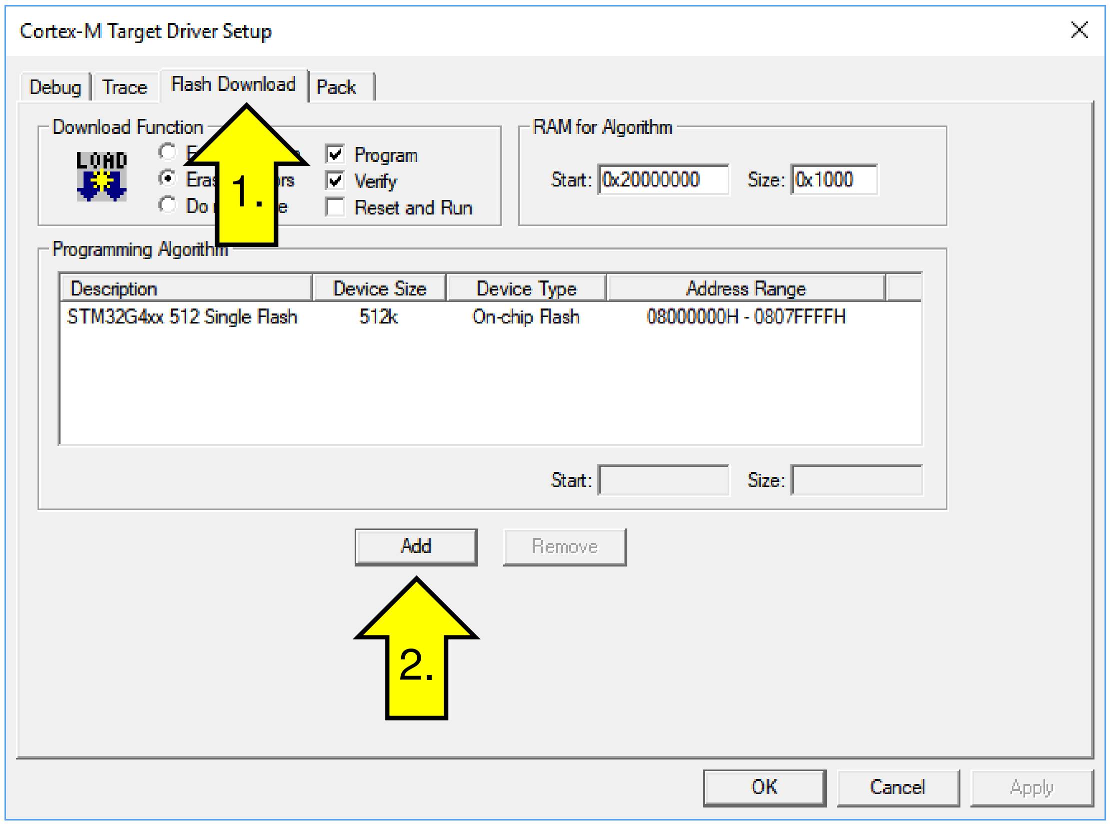
Options



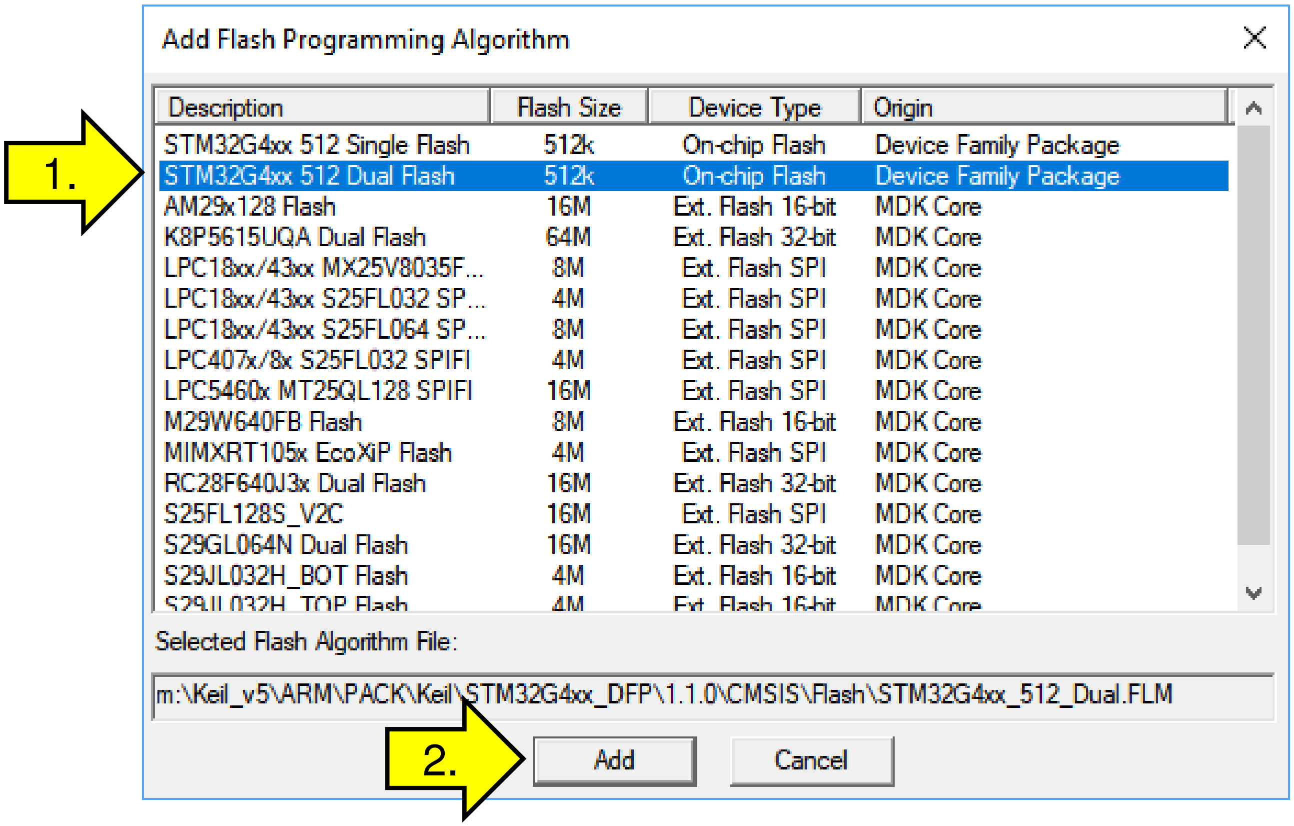
ST-Link Debugger - Settings



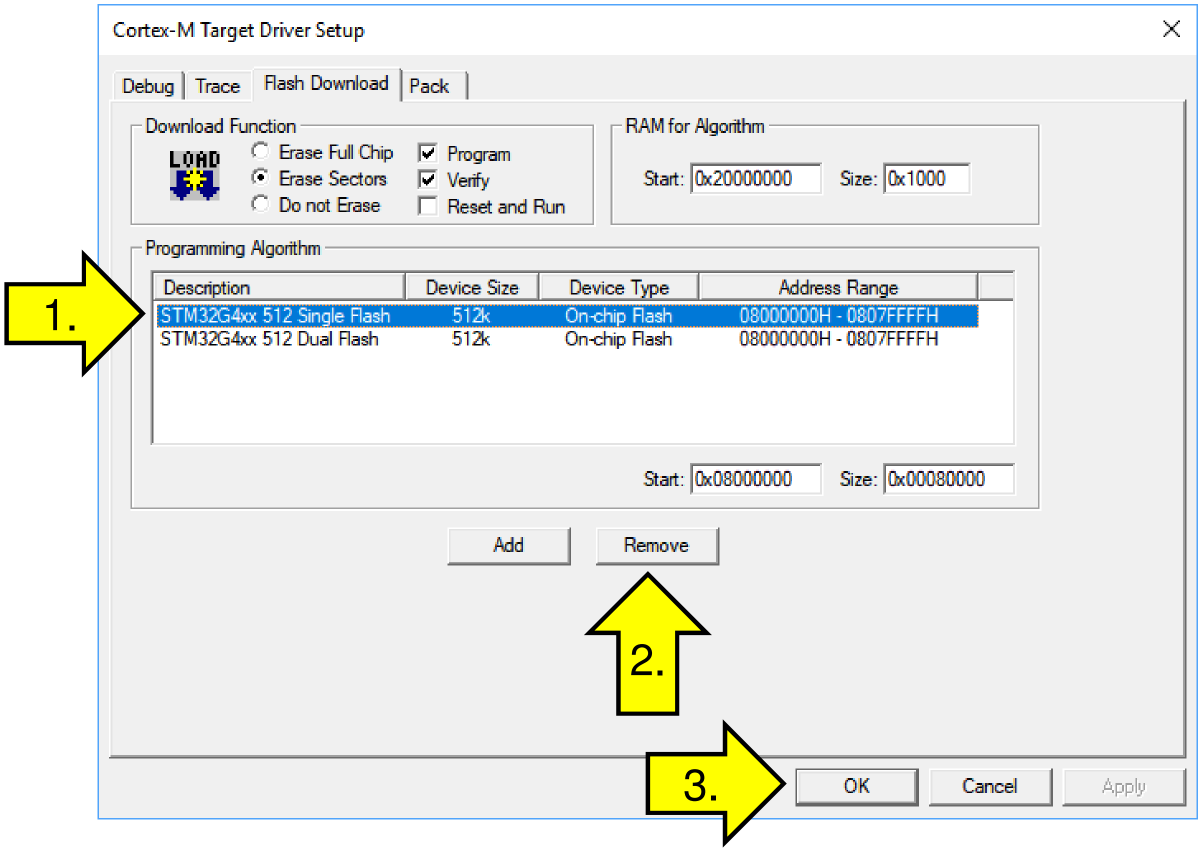
Flash Download → Add



STM32G4xx 512 Dual Flash

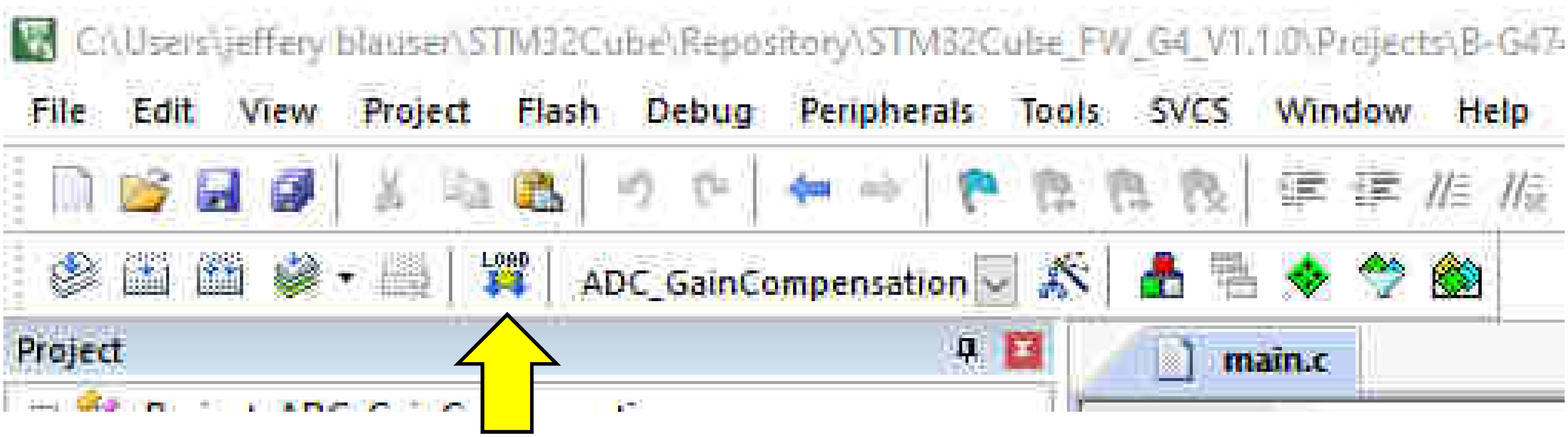


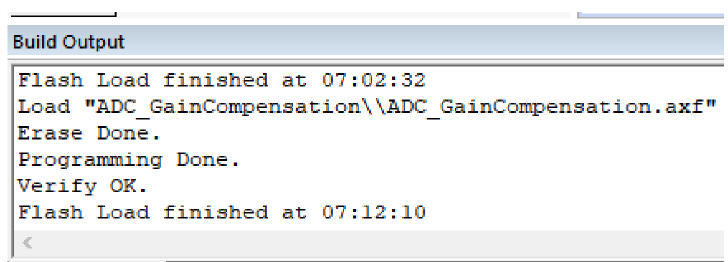
Remove Single Flash



Keil MDK-ARM

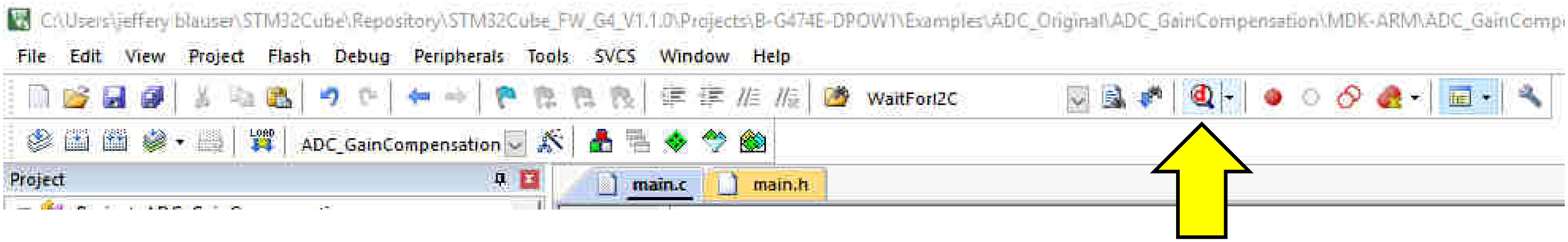
Load

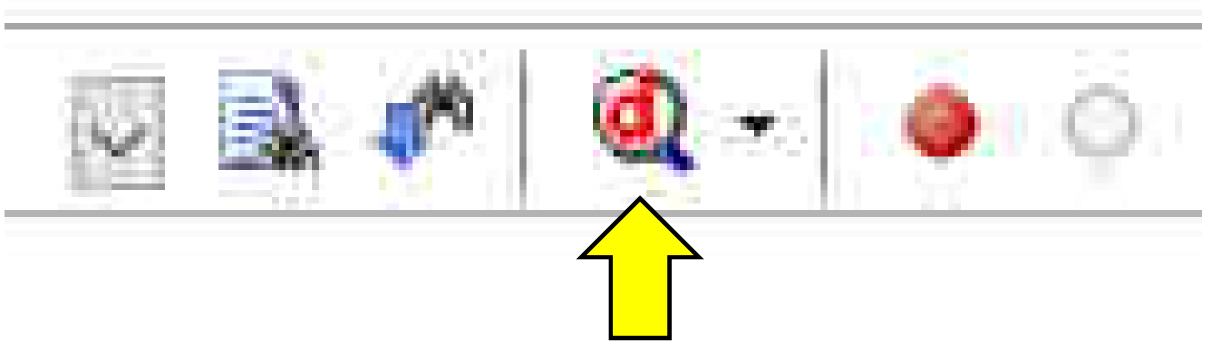



Build Output

```
Flash Load finished at 07:02:32
Load "ADC_GainCompensation\\ADC_GainCompensation.axf"
Erase Done.
Programming Done.
Verify OK.
Flash Load finished at 07:12:10
```

Debugger


File
Edit
View
Project
Flash
Debug
Peripherals
Tools
SVCS
Window
Help
WaitForI2C
ADC_GainCompensation
Project
main.c
main.h



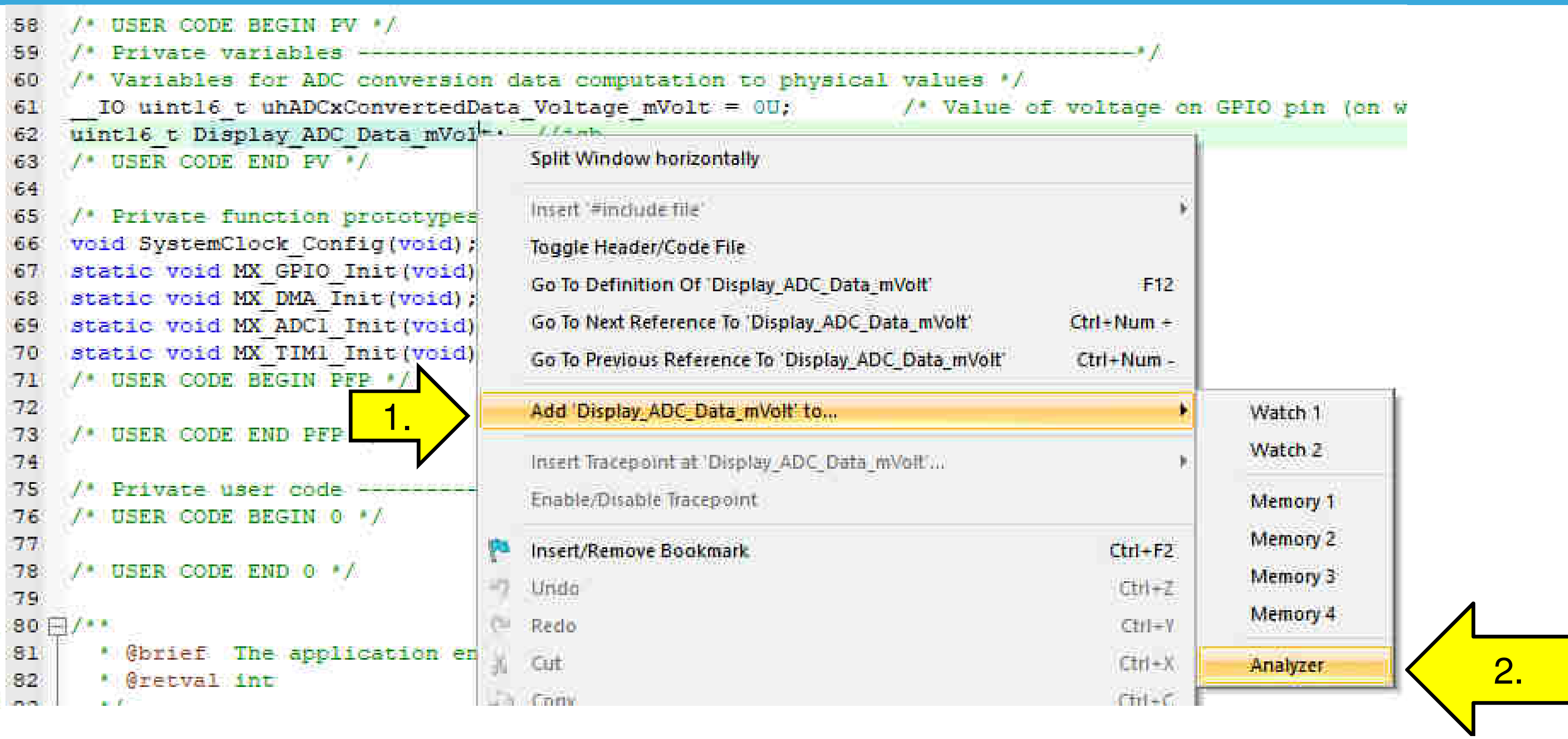
Add 2 Lines - main.c about line 153

```
148    while (1)
149    {
150      /* USER CODE END WHILE */
151
152      /* USER CODE BEGIN 3 */
153      Display_ADC_Data_mVolt=uhADCxConvertedData_Voltage_mVolt; //jsb just for display
154      HAL_Delay(1);    //jsb 1mS update rate
155    }
156    /* USER CODE END 3 */
```

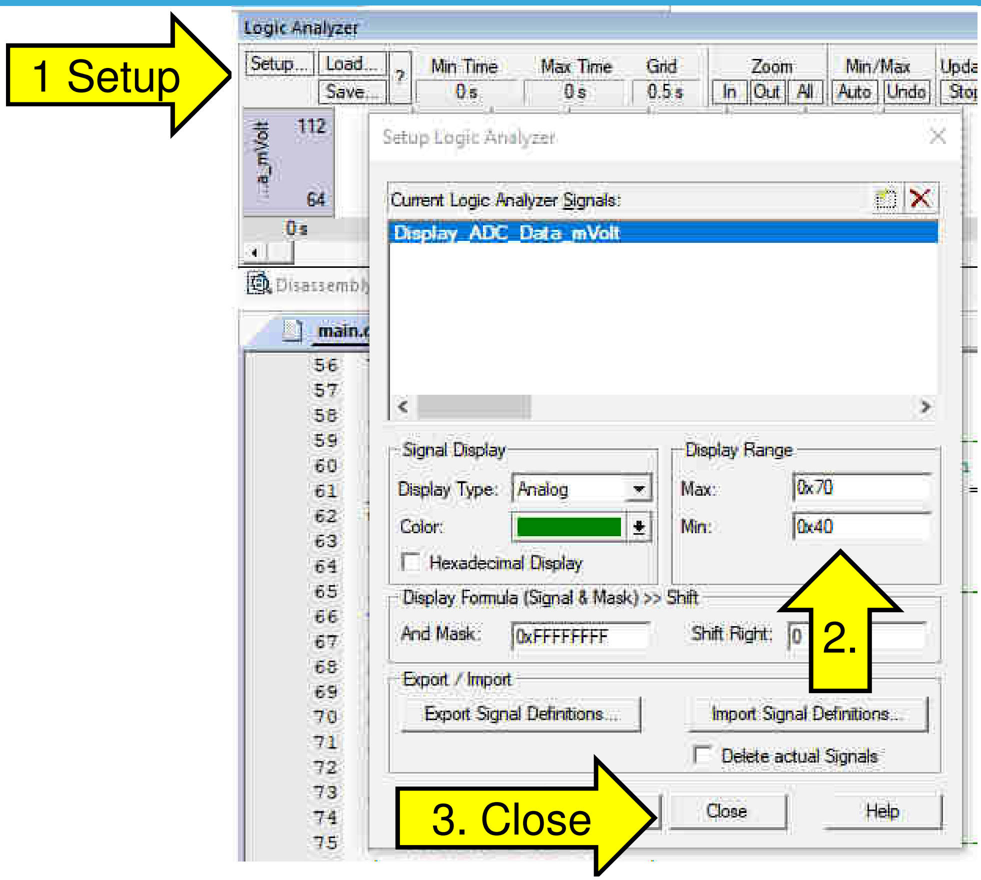
Line 153 is a global variable for use by the Logic Analyzer
Line 154 is set to 1 millisecond to pull this data from the micro to the PC for display in the Logic Analyzer.

See c:\2019_STM32G4_Workshop\HandsOn_Labs\ADC_GainCompensation
Then *CodeLinesForMain_ADC_GainCompensation.txt*

Add Line - Place variable in the Analyzer – main.c about line 62

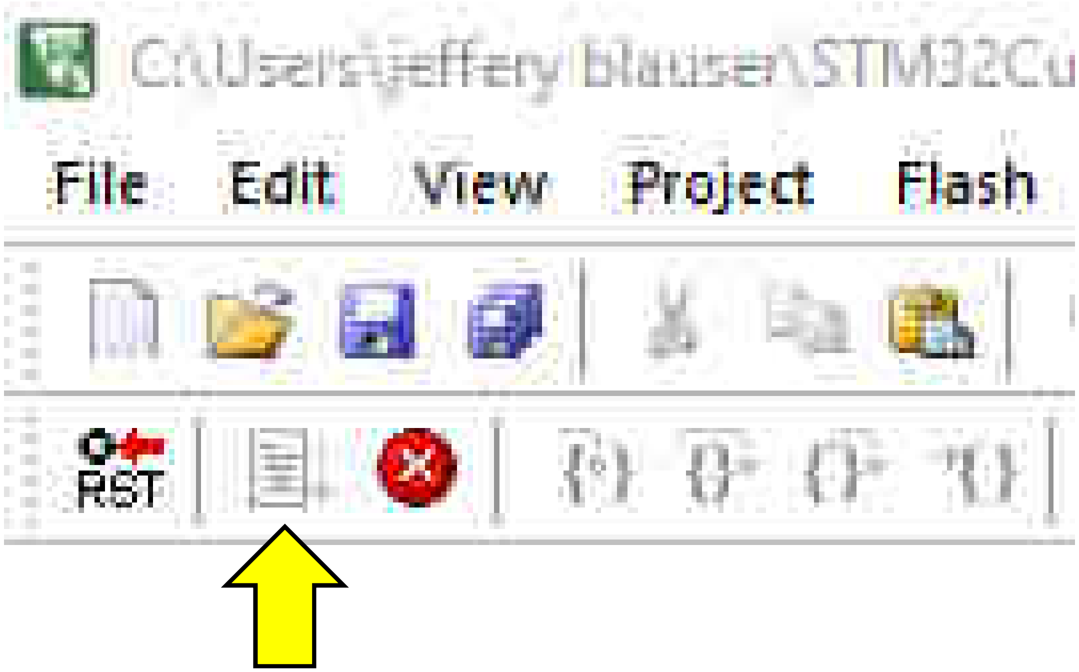


Setup – Min=0x40, Max = 0x70



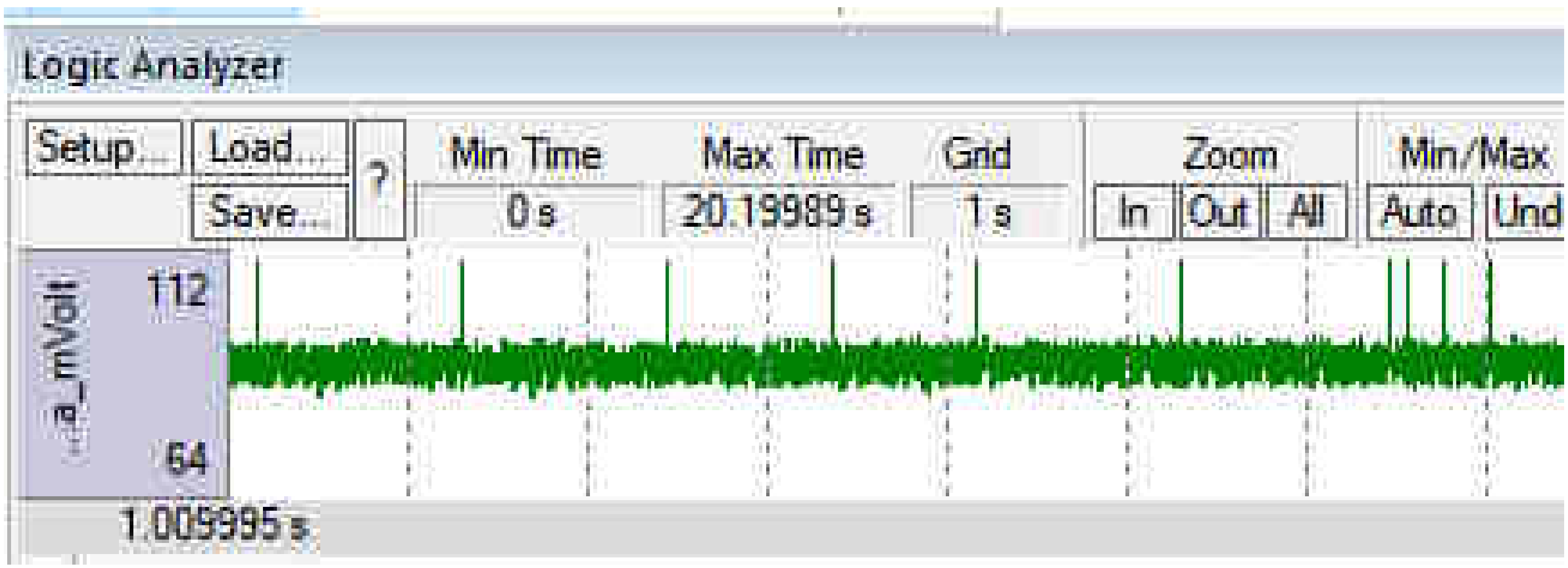
Keil MDK-ARM

RUN



Keil MDK-ARM

Logic Analyzer Output – Noise at the ADC input

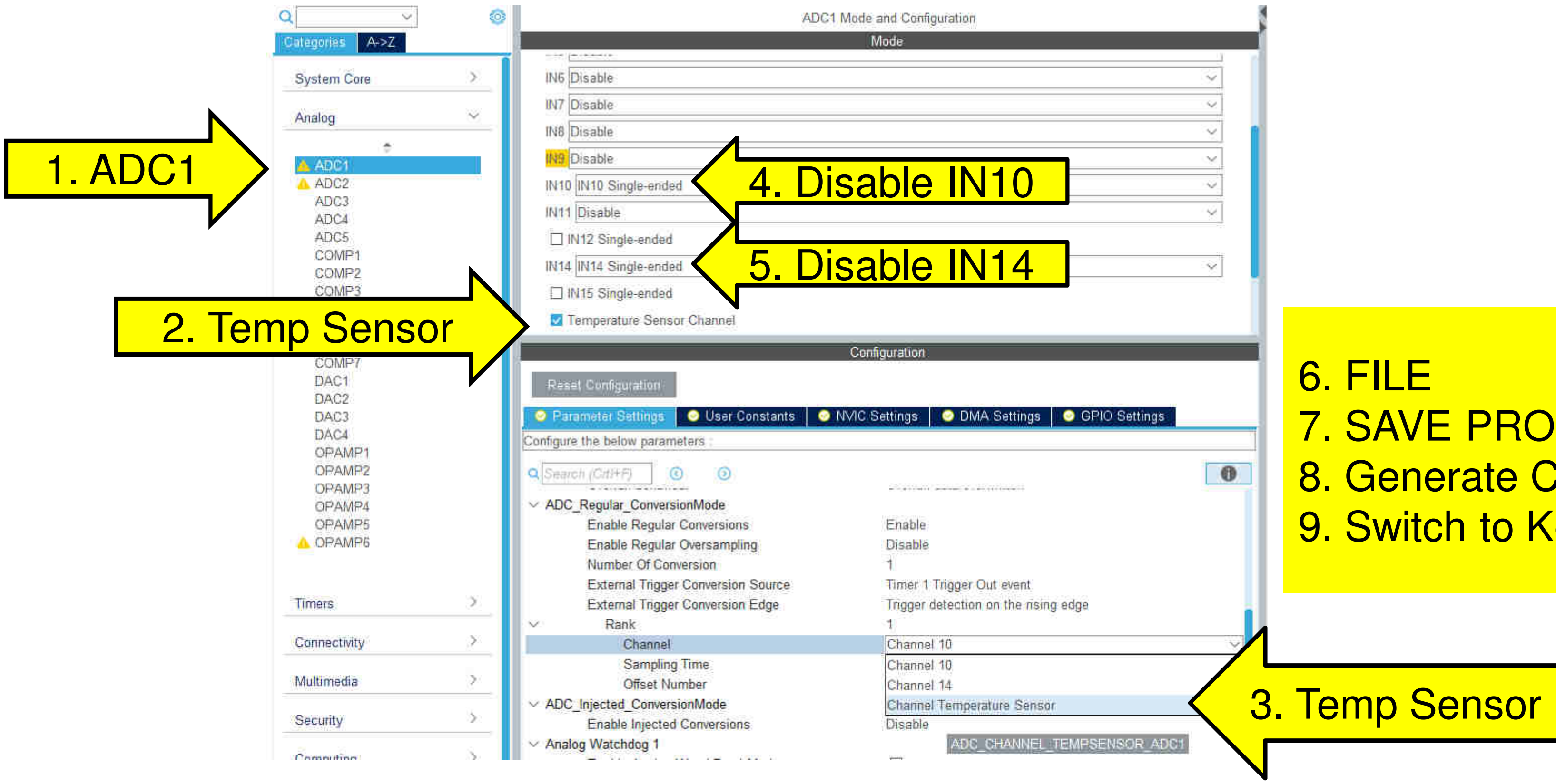


Trace Overflow

1. ADC_Trigger source = ________ {which peripheral}
2. ADC_Trigger source clock speed = ______MHz
3. ADC_Trigger source prescaler = _______
4. ADC_Trigger Speed = _____Hz
5. ADC Resolution = ______ bits
6. SWO data throughput for ADC data = __________

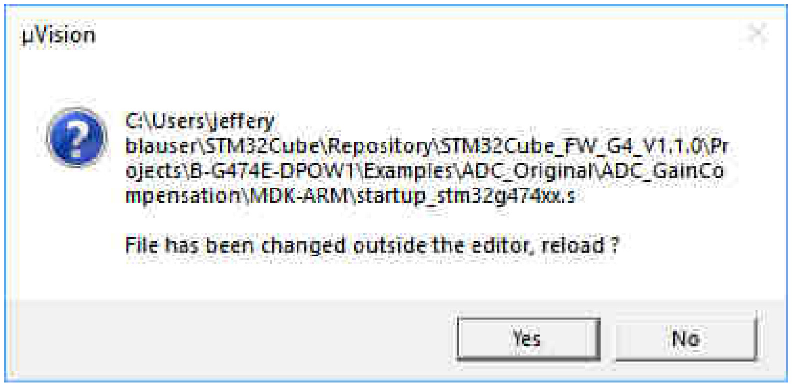
STM32CubeMx

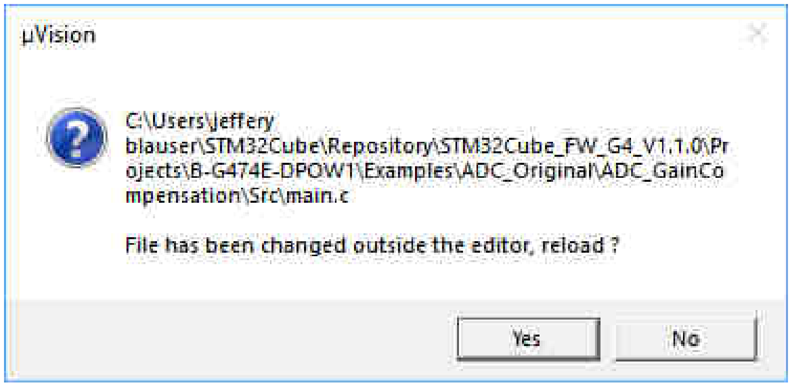
New Signal Source – Please do these steps in order



6. FILE
7. SAVE PROJECT
8. Generate Code
9. Switch to Keil Window

File Changed? Yes & Yes

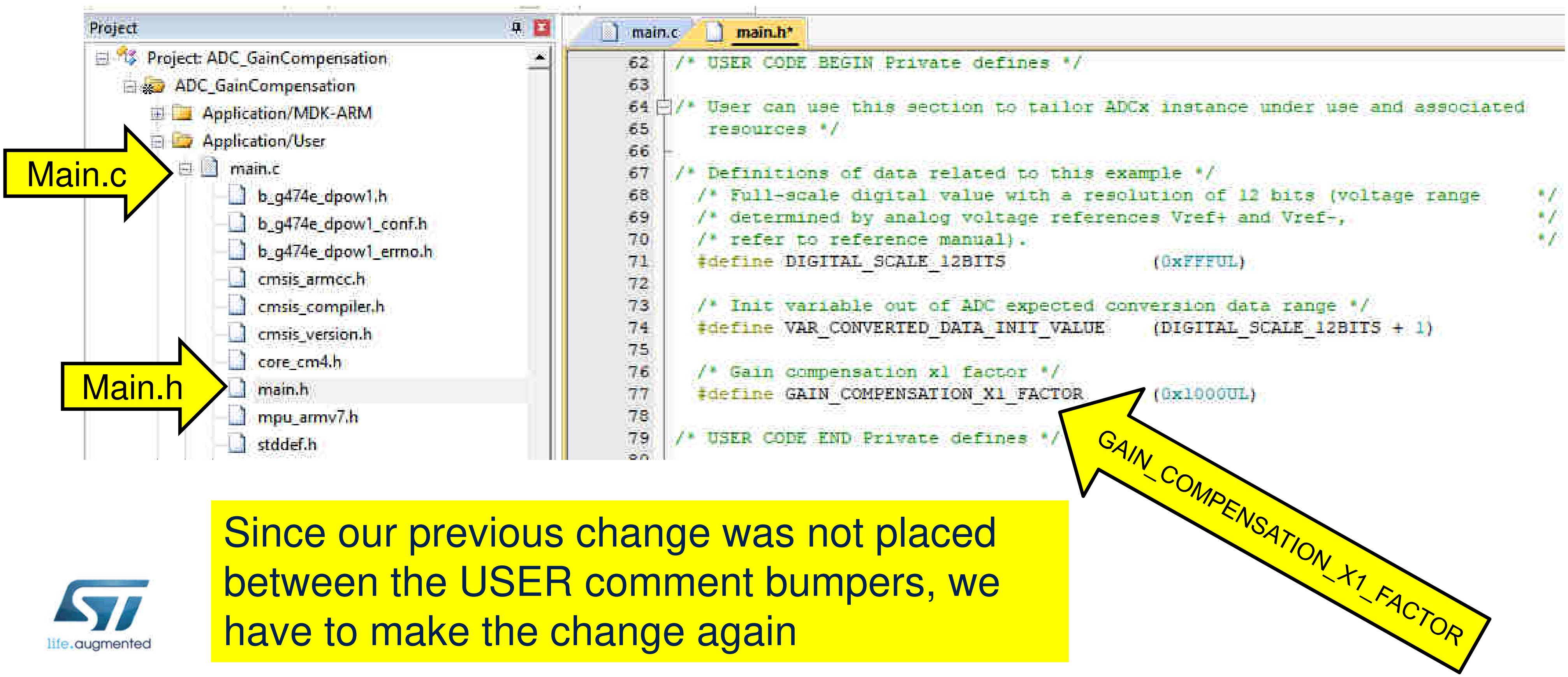




Since Keil was left open, it understands the Cube made some changes to the underlying files. You do not have to shut down Keil or the Cube, but leave them open. Click yes to each of these windows.

Keil MDK-ARM

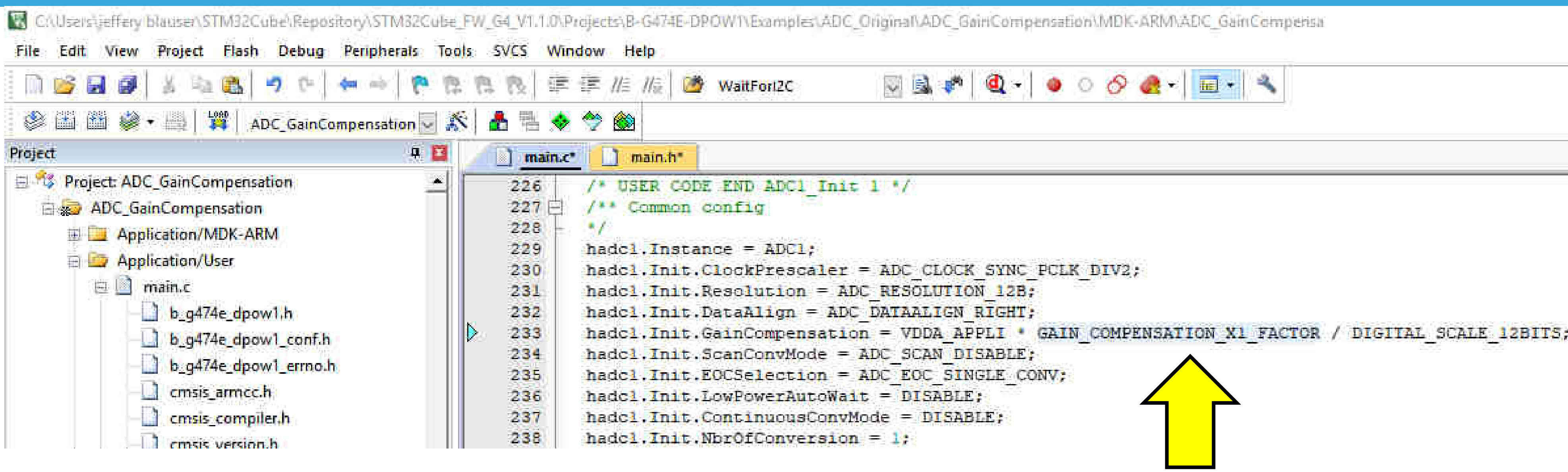
About line 77 in main.h – GAIN_COMPENSATION_X1_FACTOR



Since our previous change was not placed between the USER comment bumpers, we have to make the change again

Keil MDK-ARM

Paste into line 233 in main.c



Keil MDK-ARM

Paste into line 59-61 in main.c

```
54  /* USER CODE BEGIN PV */
55  /* Private variables ----------------------------
56  /* set to 1 after User push-button interrupt  */
57  __IO uint32_t UserButtonStatus = 0;
58
59  uint16_t  Display_DAC1DOR1;
60  uint16_t  Display_DAC3DOR2;
61  uint16_t  Display_ADC3IN1;
62  uint32_t *ts_cal1_p = (uint32_t*)0x1FFF75A8;
63  uint32_t *ts_cal2_p = (uint32_t*)0x1FFF75CA;
64  uint16_t TS_CAL1;
65  uint16_t TS_CAL2;
66
```

Keil MDK-ARM

Paste into line 62-66 in main.c

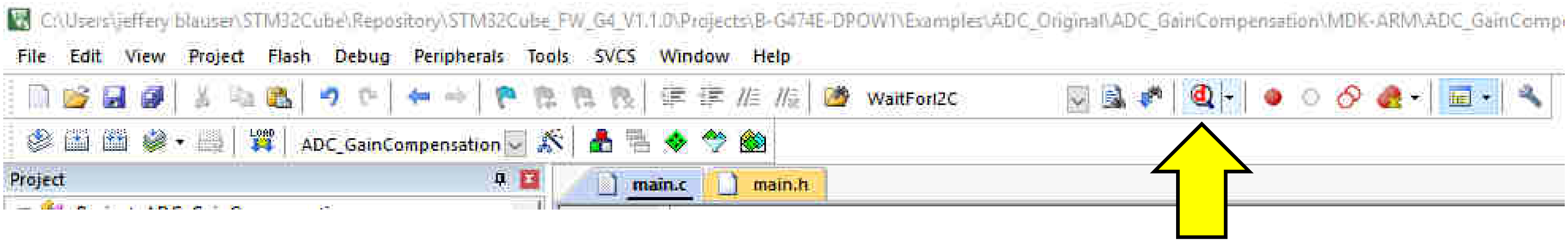
```
54 /* USER CODE BEGIN PV */
55 /* Private variables ---------------------------------------------------------
56 /* set to 1 after User push-button interrupt  */
57 __IO uint32_t UserButtonStatus = 0;
58
59 uint16_t  Display_DAC1DOR1;
60 uint16_t  Display_DAC3DOR2;
61 uint16_t  Display_ADC3IN1;
62 uint32_t *ts_cal1_p = (uint32_t*)0x1FFF75A8;
63 uint32_t *ts_cal2_p = (uint32_t*)0x1FFF75CA;
64 uint16_t TS_CAL1;
65 uint16_t TS_CAL2;
66 uint16_t Display_TemperatureC;
```

Keil MDK-ARM

Paste into line 131 in main.c

```
126   /* USER CODE BEGIN 2 */
127   /* Initialize the User push-button.
128      It is used for changing the gain */
129   BSP_PB_Init(BUTTON_USER, BUTTON_MODE_EXTI);
130
131   /* Establish the Temperature Sensor Calibration Values */
132   TS_CAL1 = *ts_cal1_p & 0xFFFF;    //Read only the two LSBs
133   TS_CAL2 = *ts_cal2_p & 0xFFFF;    //Read only the two LSBs
134
```

Debugger

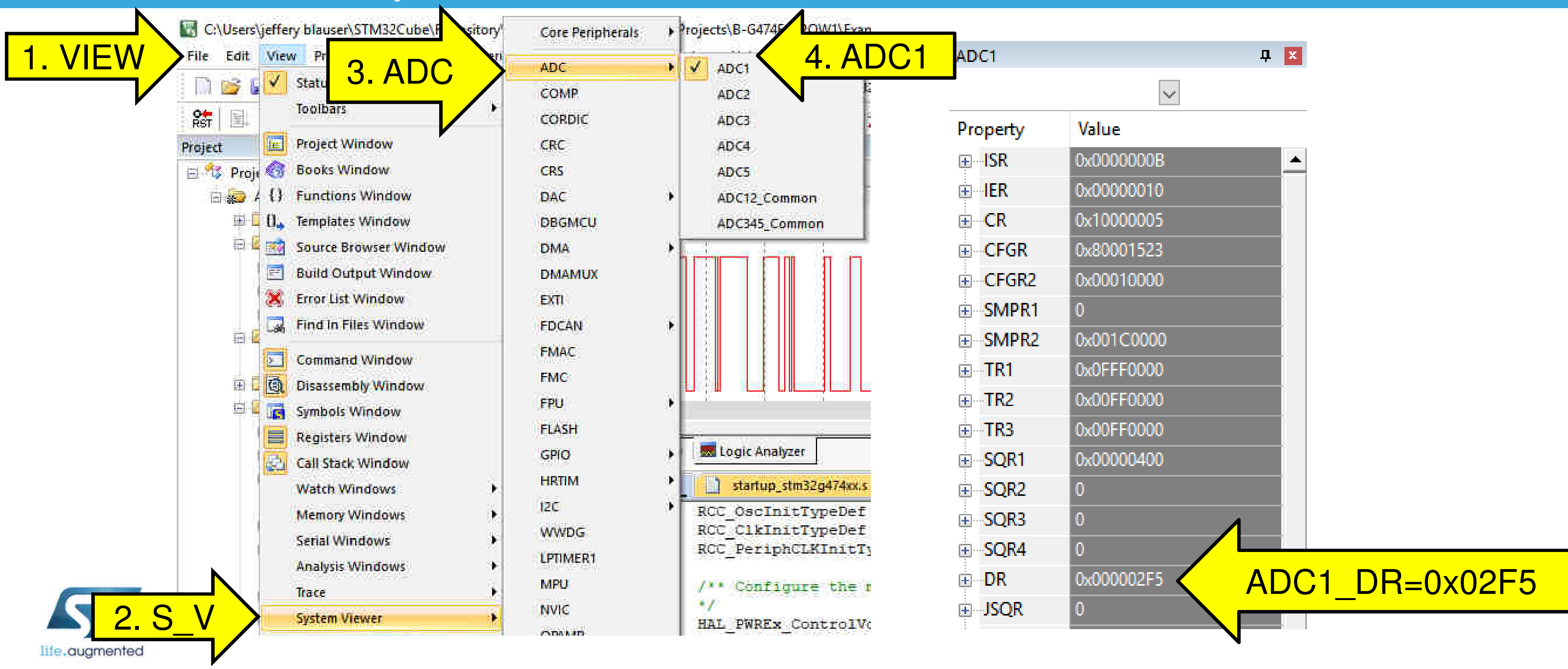

C:\Users\jeffery.blauser\STM32Cube\Repository\STM32Cube_FW_G4_V1.1.0\Projects\B-G474E-DPOW1\Examples\ADC_Original\ADC_GainCompensation\MDK-ARM\ADC_GainComp
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
WaitForI2C
ADC_GainCompensation
Project
main.c
main.h

RUN


C:\Users\jeffery blauser\STM32Cu
File Edit View Project Flash
RST

Keil MDK-ARM

VIEW→System Viewer→ADC→ADC1, then ADC1_DR = 0x02F5



life.augmented

Internal Temperature Sensor

Temperature Calculation –RM0440, section 20.4.31

$$Temperature(°C) = \frac{TS_{Cal2Temp} - TSCal1Temp}{TS_{Cal2} - TS\_Cal1}\ X\ (TS\_Data\text{-}TS\_Cal1) + 30°C$$

Table 5. Temperature sensor calibration values

Calibration value name	Description	Memory address
TS_CAL1	TS ADC raw data acquired at a temperature of 30 °C (± 5 °C), $V_{DDA} = V_{REF+}$ = 3.0 V (± 10 mV)	0x1FFF 75A8 - 0x1FFF 75A9
TS_CAL2	TS ADC raw data acquired at a temperature of 110 °C (± 5 °C), $V_{DDA} = V_{REF+}$ = 3.0 V (± 10 mV)	0x1FFF 75CA - 0x1FFF 75CB

$$Temperature\ (°C) = \frac{110 - 30}{1373 - 795}\ x\ (757 - 795) + 30 = 24.7°C$$

life.augmented

Internal Temperature Sensor

New Gain Factor

$$Temperature\ (°C) = \frac{110-30}{1373-795}\ x\ (757 - 795) + 30 = 24.7°C$$

Our new Gain factor is $\frac{110-30}{1373-795} = 0.238$

We know that the ratio for Gain Compensation is 0-3.999756 for 0-16383 span.

$$\frac{0.238}{3.99756} = \frac{GCompCoeff}{16383}$$

GCompCoeff = 975 = 0x03CF

Keil MDK-ARM

Main.h - FILE→SAVE→BUILD

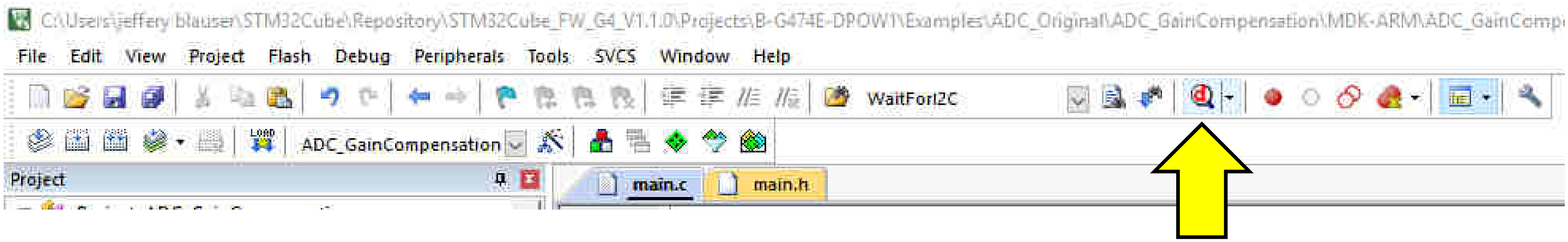
```
/* Gain compensation x1 factor */
#define GAIN_COMPENSATION_X1_FACTOR        (0x03CFUL)
```

FROM:
0x1000UL

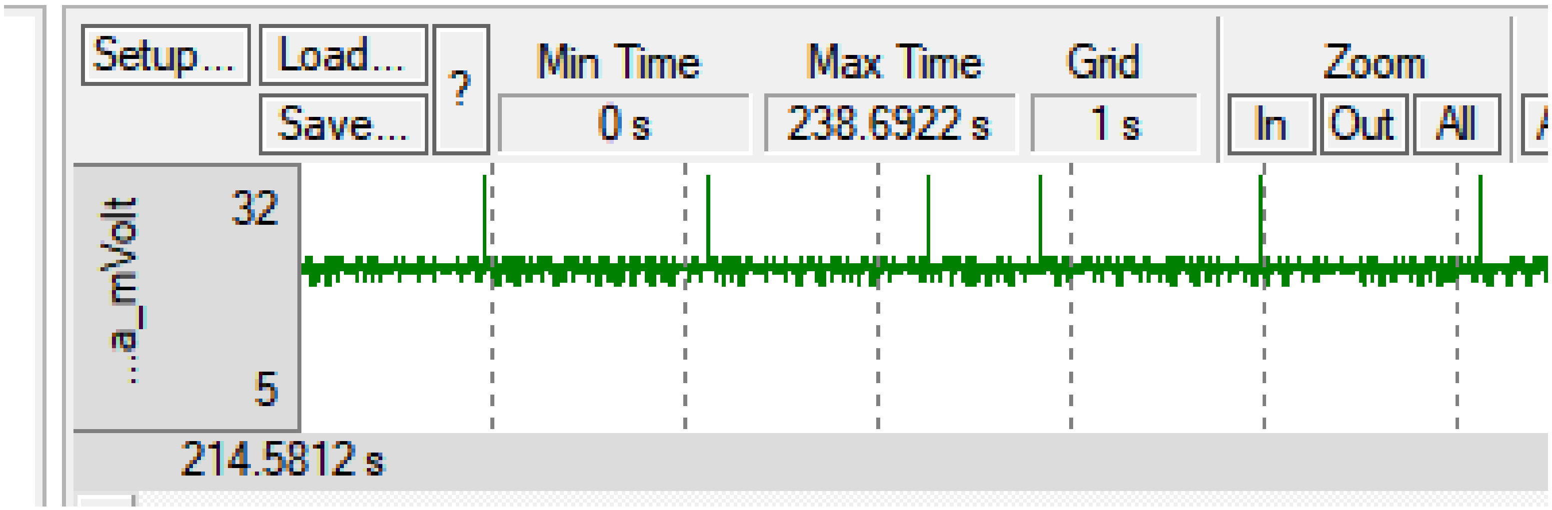
TO:
0x03CFUL

Keil MDK-ARM

Debugger



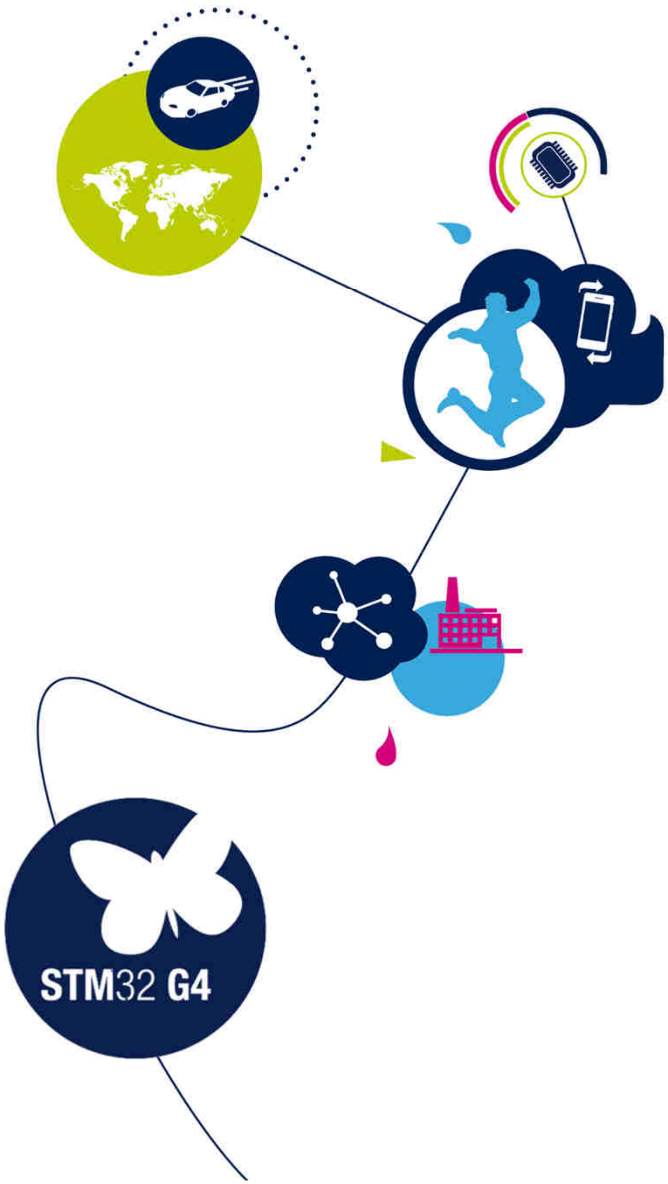
Logic Analyzer Output



Results are in the mid-20°C with just a bit of noise

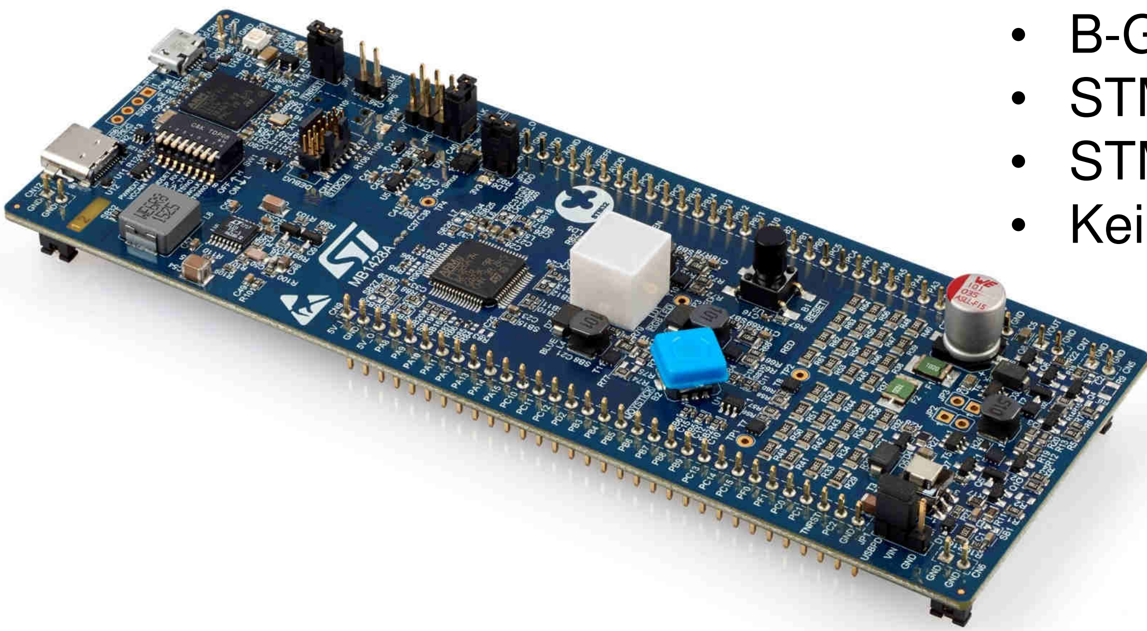
STM32G4 mixed-signal MCU workshop

Hands On:
DAC Signal Generation



life.augmented

Systems Check



- USB cable
- B-G474E-DPOW1 Discovery Board
- STM32G4 Family Cube Library
- STM32CubeMx
- Keil MDK-ARM

Digital to Analog Converter

G4 DAC

- 3 external outputs:
  - DAC1-OUT1 and OUT2, 1Msps
  - DAC2-OUT1, 1Msps
- 4 internal outputs:
  - DAC3 and DAC4
  - OUT1 and OUT2, 15Msps
- Double data DMA
- Sample & Hold
- Left or Right data alignment
- Signed or unsigned data input format {Q1.15, Q1.11, Q1.7}

Internally connected to:

- Voltage reference
- External Triggers
- Comparator inputs
- Op Amp inputs
- Multiple Timers (+ HRTimer)
- DMA

Digital to Analog Converter

G4 DAC Sample and Hold

DAC conversion during the “sample and hold“ mode has three phases:
1. Sampling phase: the “sample and hold“ element is charged with the desired voltage.
2. Holding phase: the DAC output is tri-stated (High-Z) to maintain the “sample and hold“ element’s stored
electrical charge.
3. Refresh phase: due to leakage coming from several sources, a refresh phase is essential to maintain the output voltage at the desired value (+/-LSB).

life.augmented

Digital to Analog Converter

Dual Conversion from DMA

We will use the DAC dual channel mode with DMA to generate signals on both DAC channels at the same time.

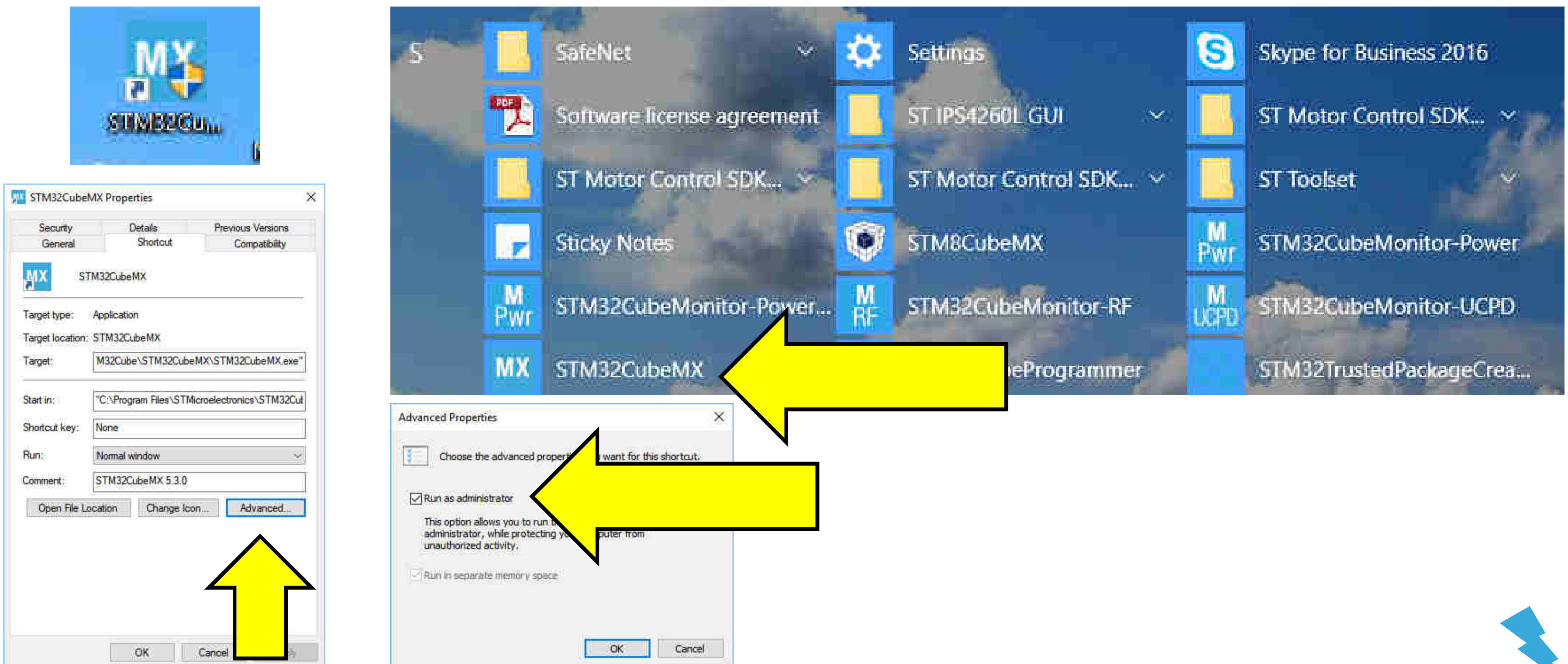
DAC conversions are made with a timer trigger and data fed from DMA.

The timer is configured to trigger the DAC channels at about 1kHz.
DMA is feeding both channels at the time.
Channel 1 pattern is ~0V/~1.65V/~3.3V/~1.65V
Channel 2 pattern is ~1.65V/~3.3V/~1.65V/~0V

We will use the built in Logic Analyzer today instead of an o’scope on the DACs

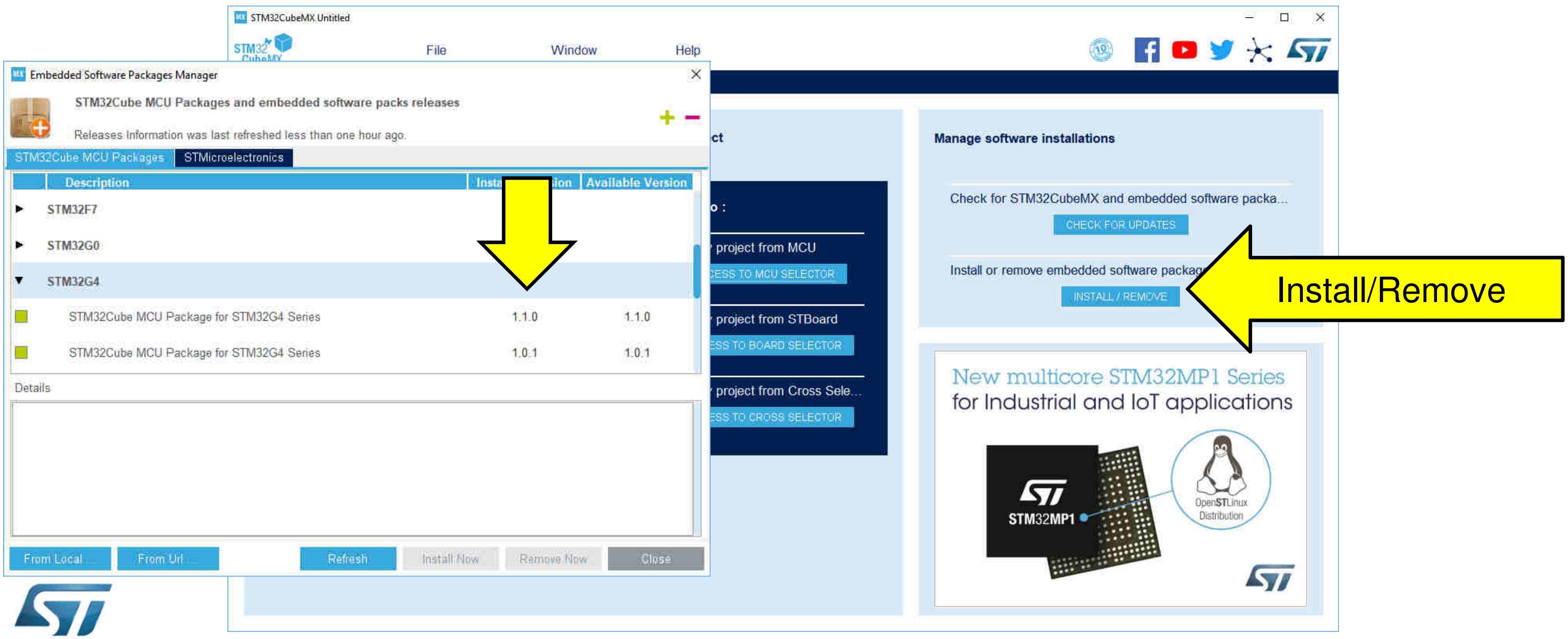
STM32CubeMx

Start up



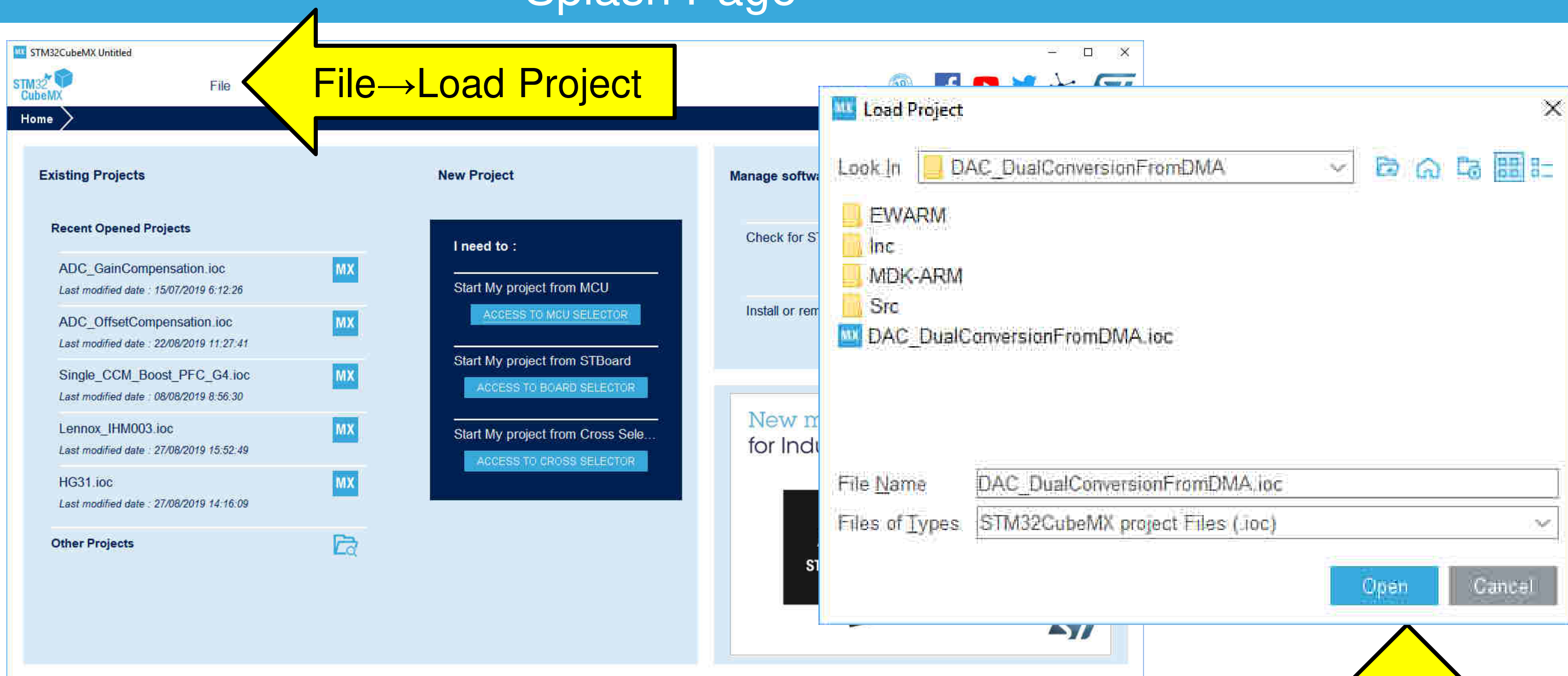
STM32CubeMx

Splash Page – G4 Library Installed?



STM32CubeMx

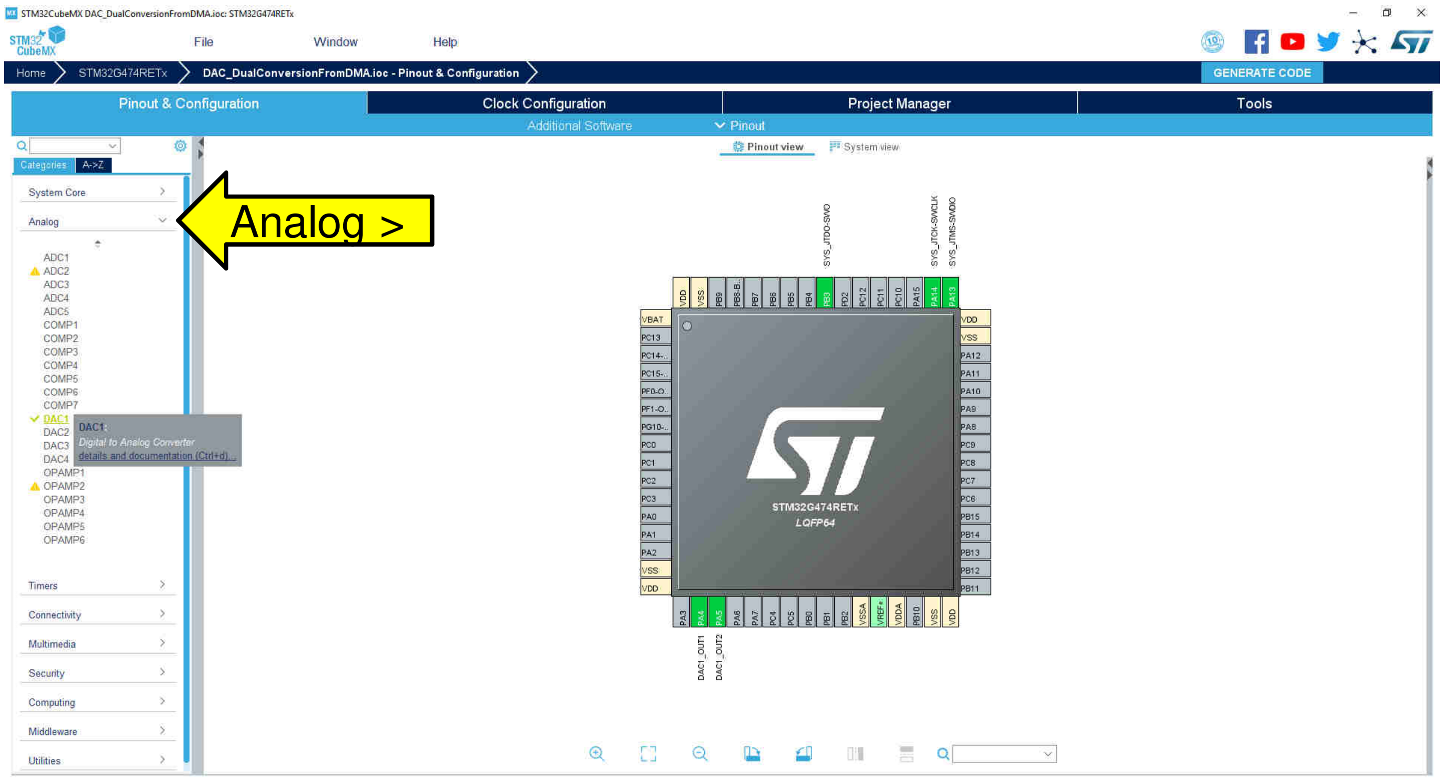
Splash Page



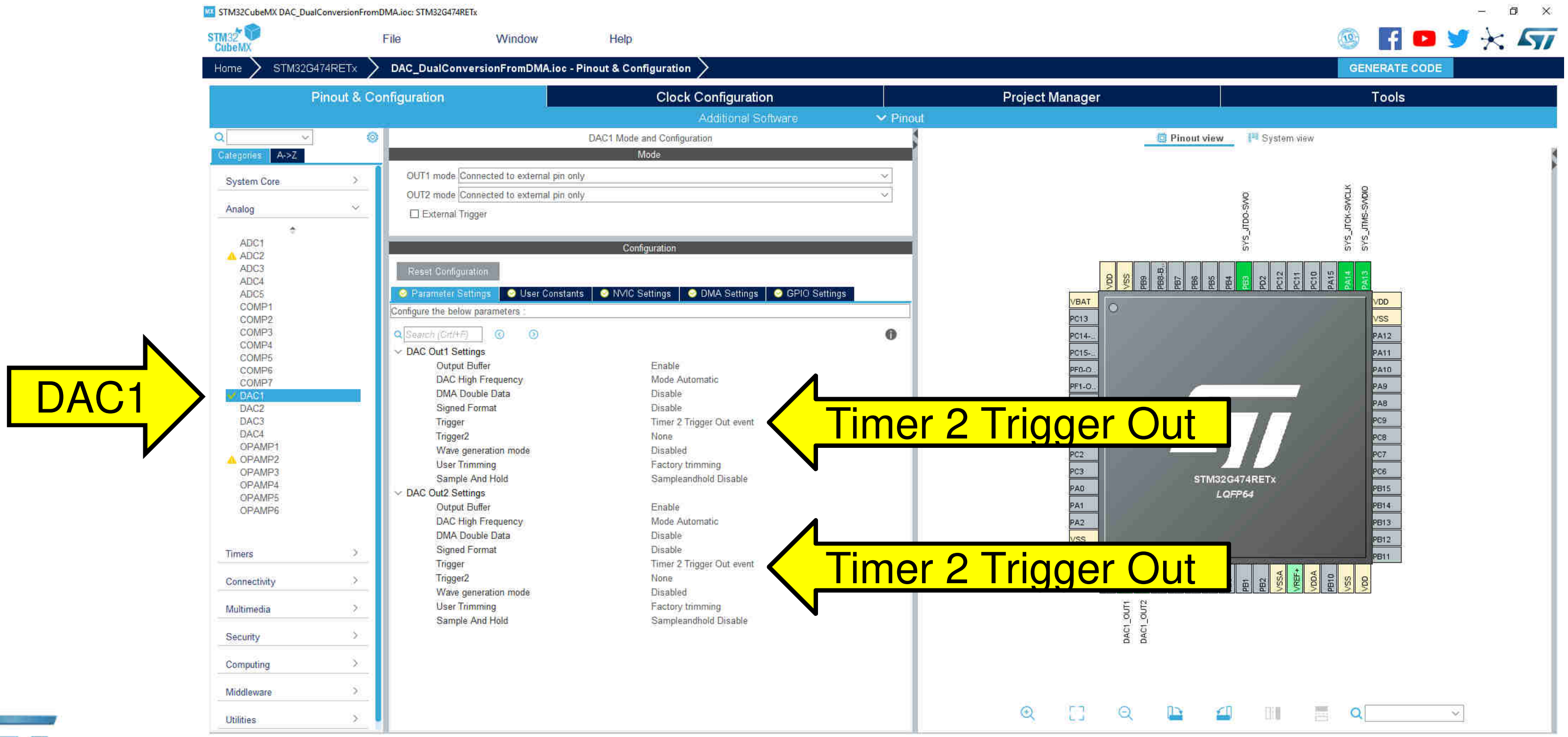
C:\...\STM32Cube\Repository\STM32Cube_FW_G4_V1.1.0\Projects\
B-G474E-DPOW1\Examples\DAC\DAC_DualConversionFromDMA

STM32CubeMx

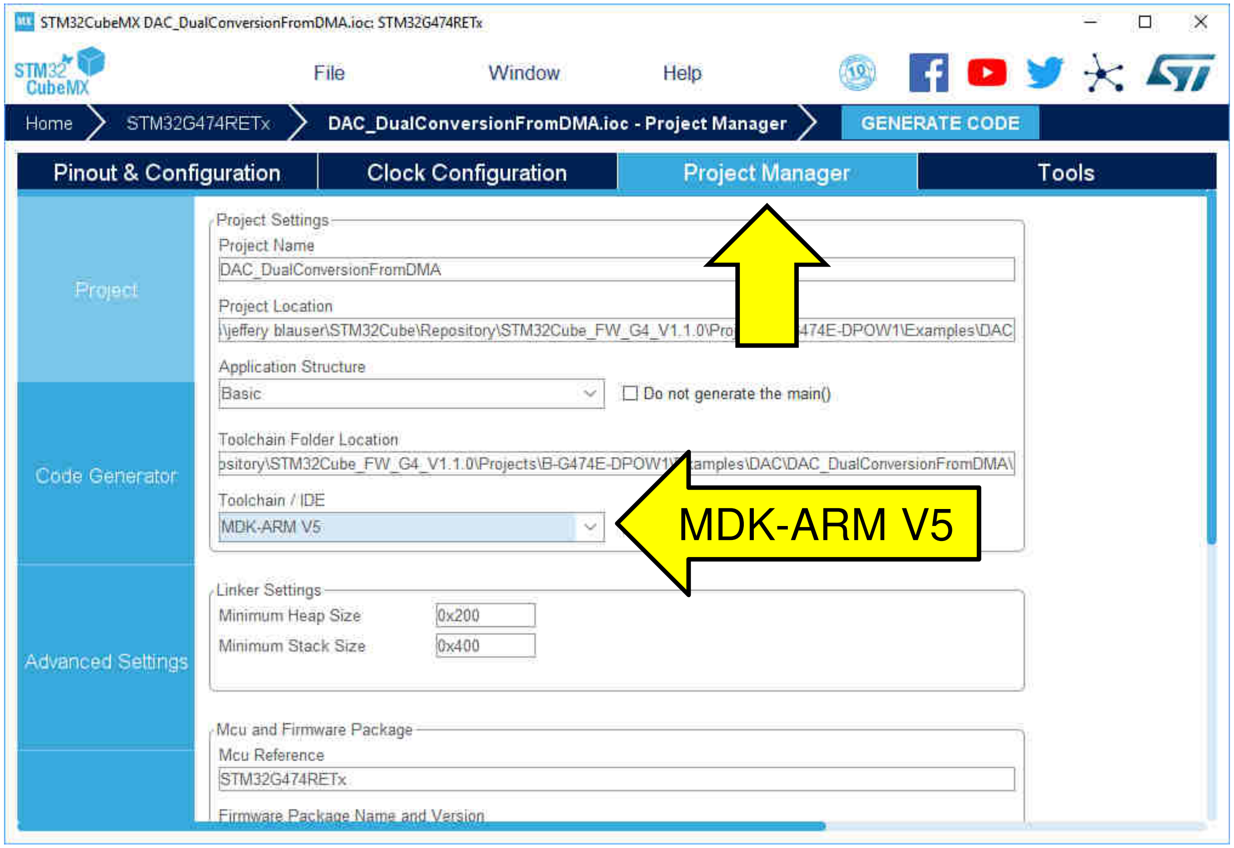
Graphic Design Representation



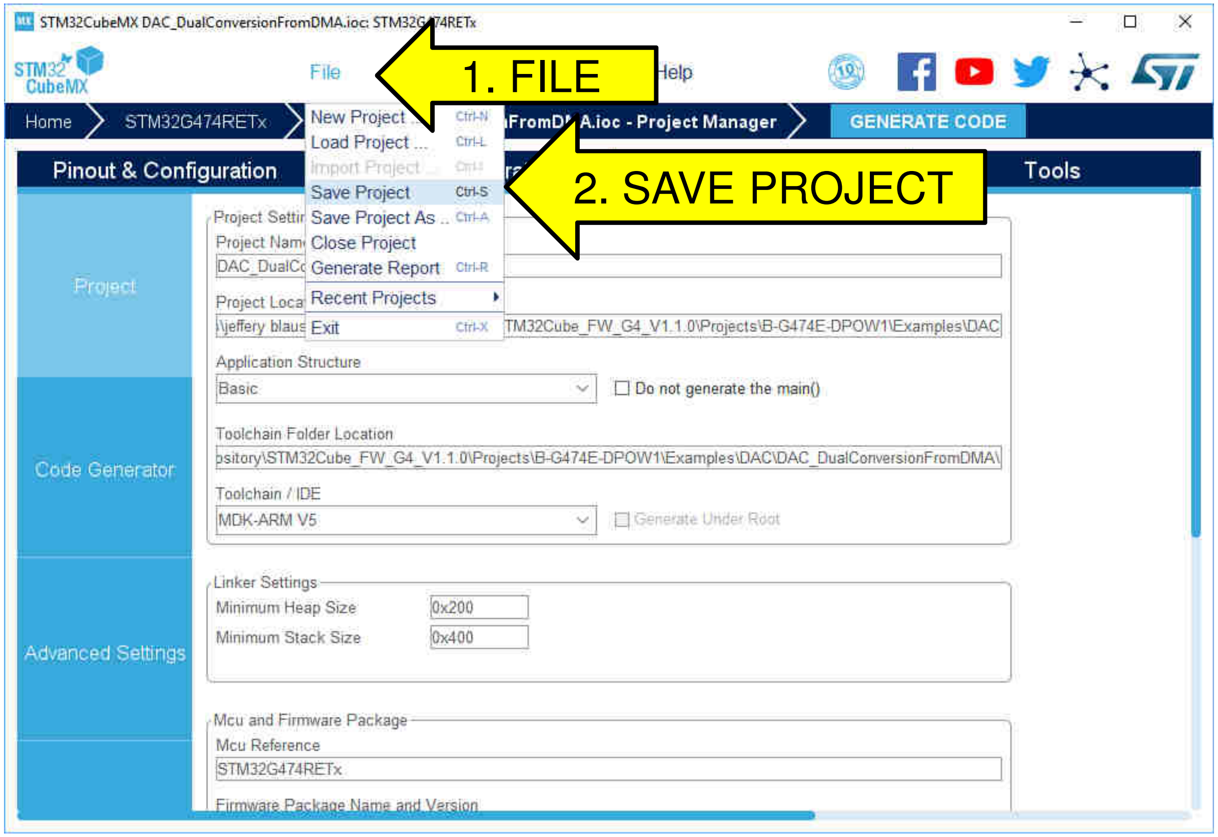
DAC1, Timer 2


DAC1
Timer 2 Trigger Out
Timer 2 Trigger Out
DAC1 Mode and Configuration
OUT1 mode Connected to external pin only
OUT2 mode Connected to external pin only
DAC Out1 Settings
Output Buffer Enable
DAC High Frequency Mode Automatic
DMA Double Data Disable
Signed Format Disable
Trigger Timer 2 Trigger Out event
Trigger2 None
Wave generation mode Disabled
User Trimming Factory trimming
Sample And Hold Sampleandhold Disable
DAC Out2 Settings
Output Buffer Enable
DAC High Frequency Mode Automatic
DMA Double Data Disable
Signed Format Disable
Trigger Timer 2 Trigger Out event
Trigger2 None
Wave generation mode Disabled
User Trimming Factory trimming
Sample And Hold Sampleandhold Disable
STM32G474RETx
LQFP64
GENERATE CODE

Project Manager – Keil MDK-ARM V5



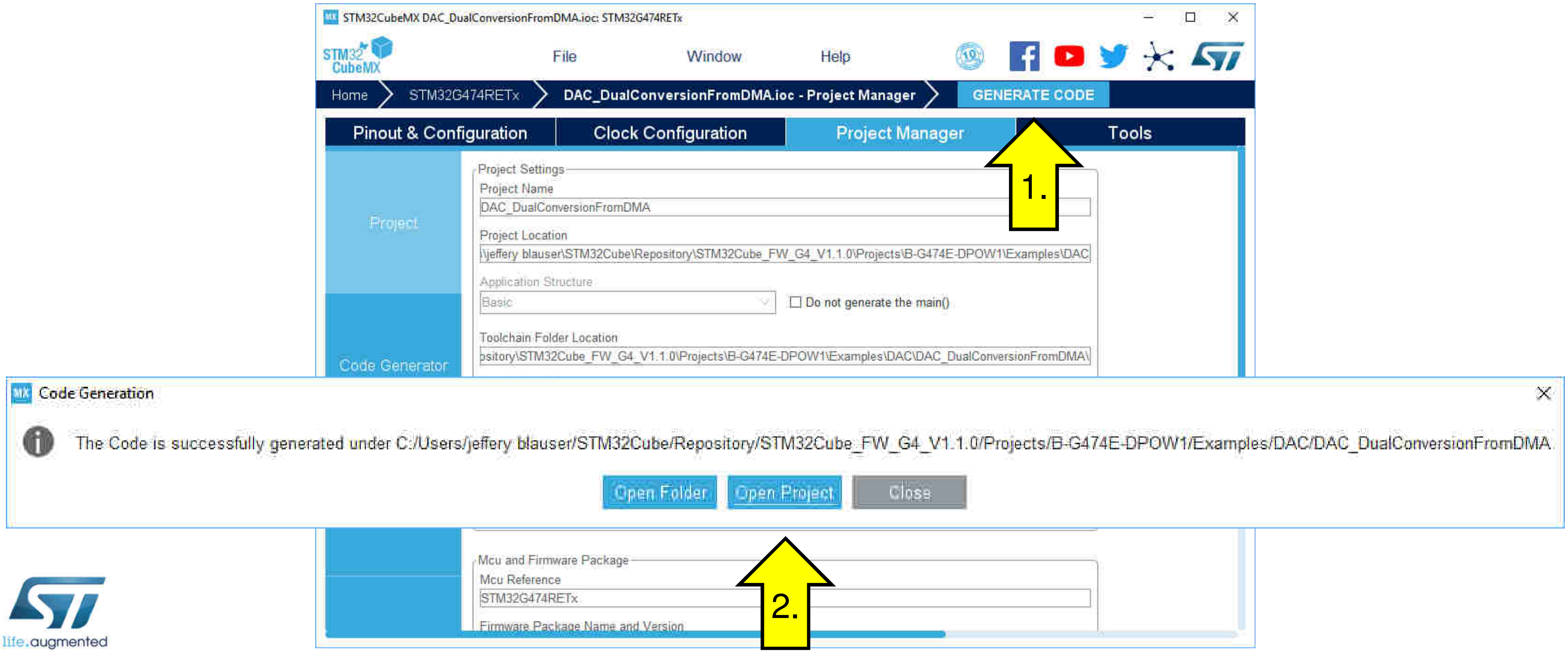
FILE→SAVE PROJECT



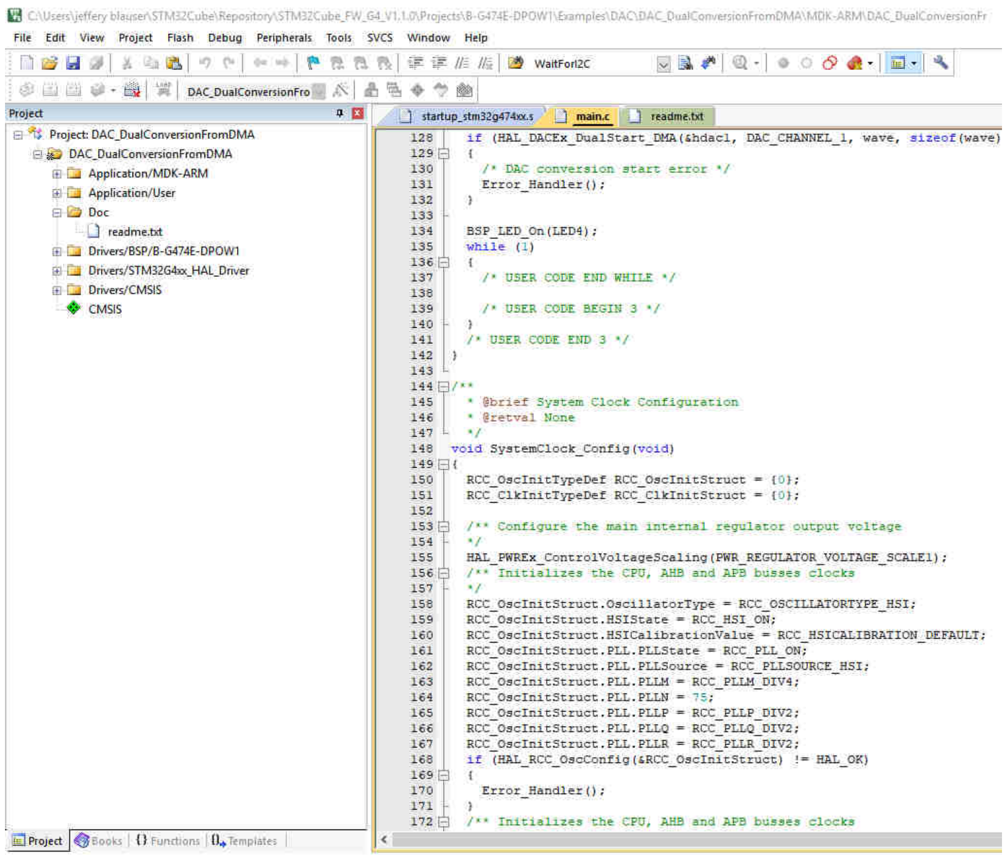
life.augmented

CubeProgrammer

Generate Code

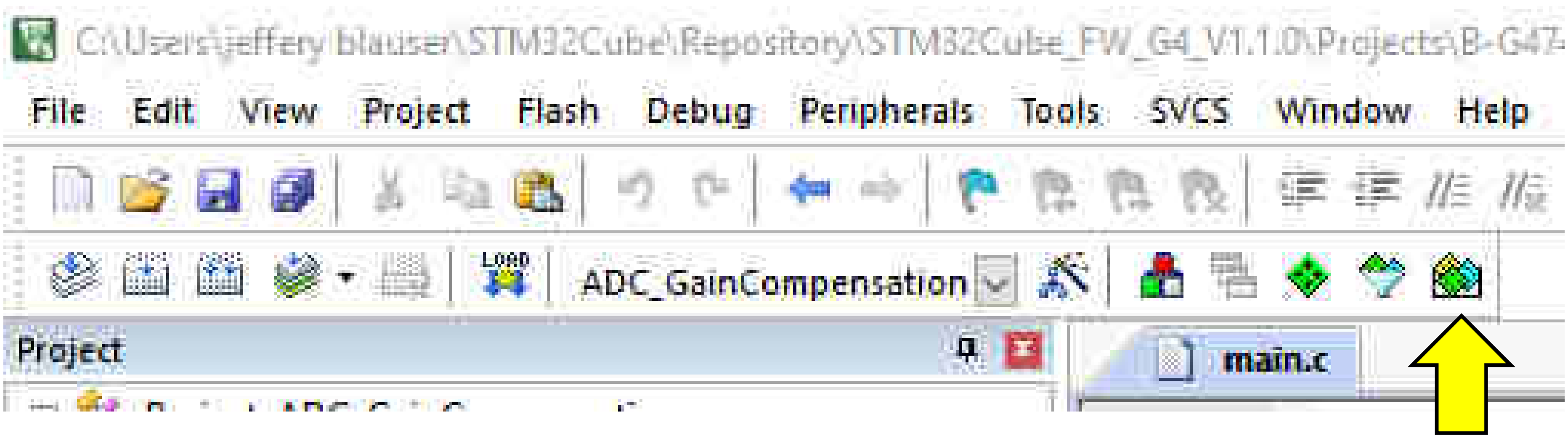


Keil Start up page



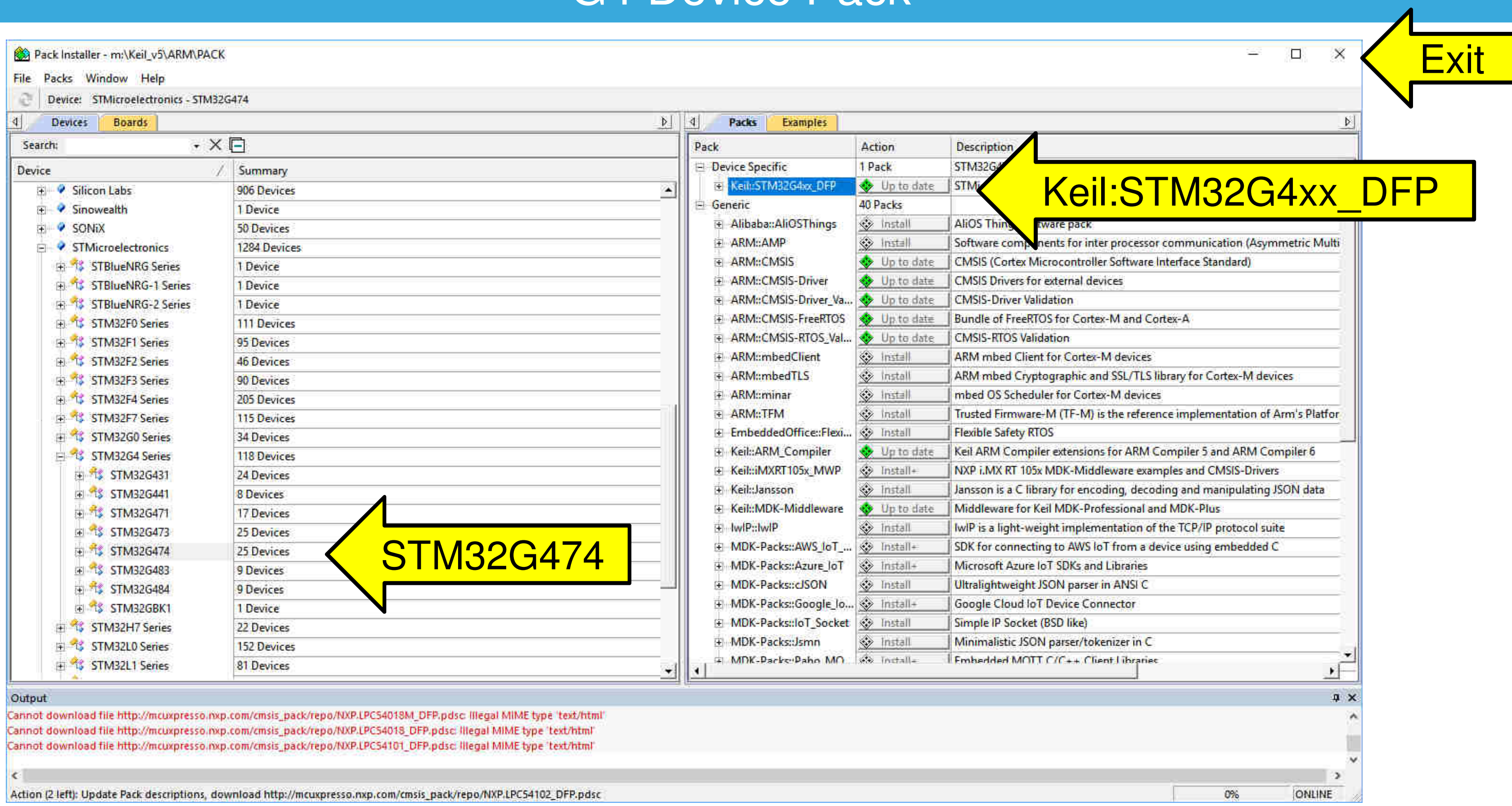
Keil MDK-ARM

Check the Device Pack



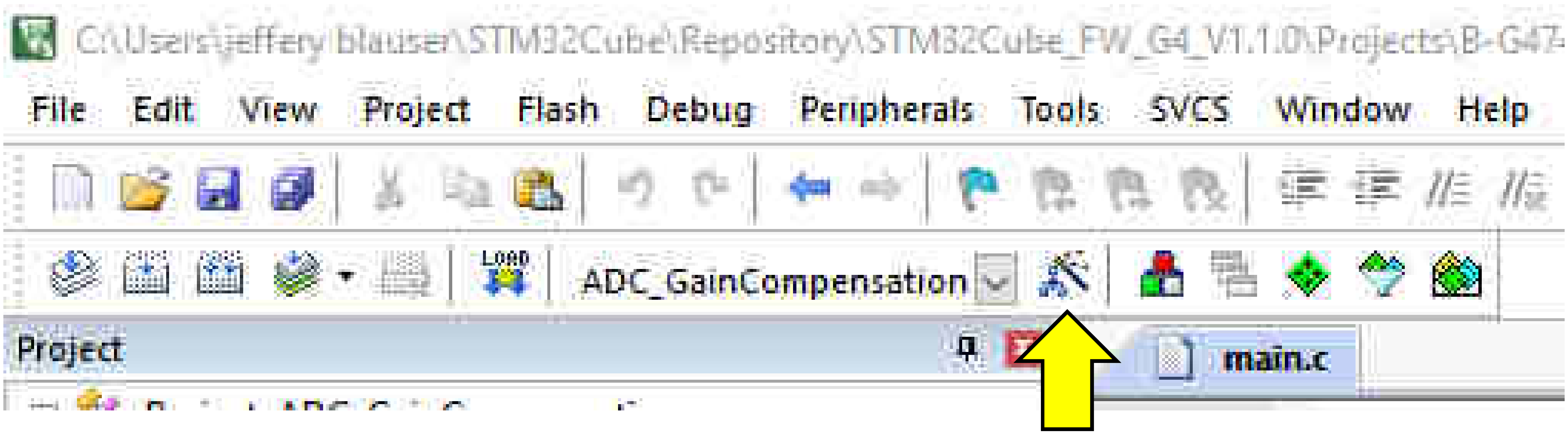
Keil MDK-ARM

G4 Device Pack



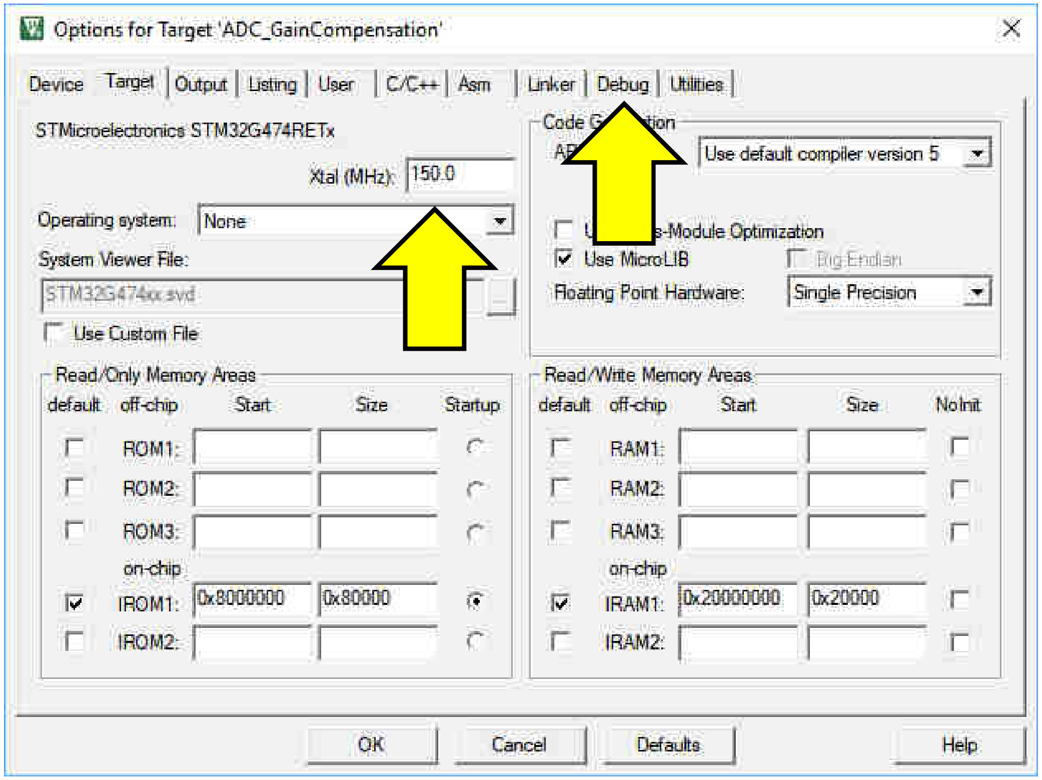
Keil MDK-ARM

Check the Options

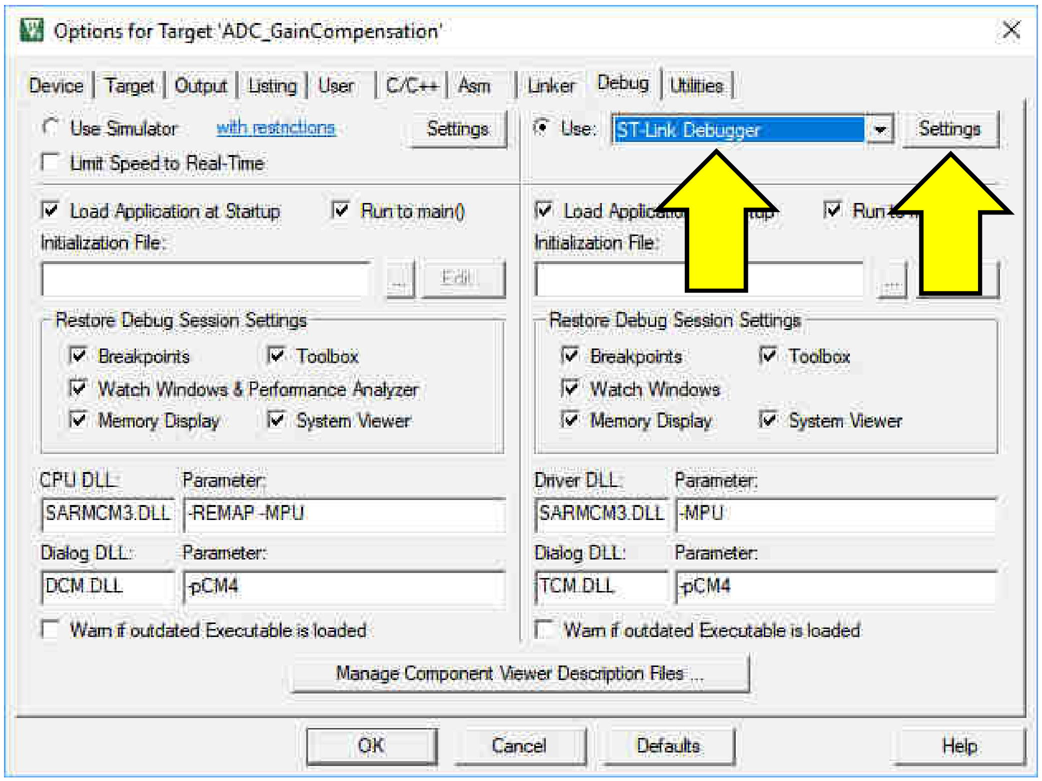


Keil MDK-ARM

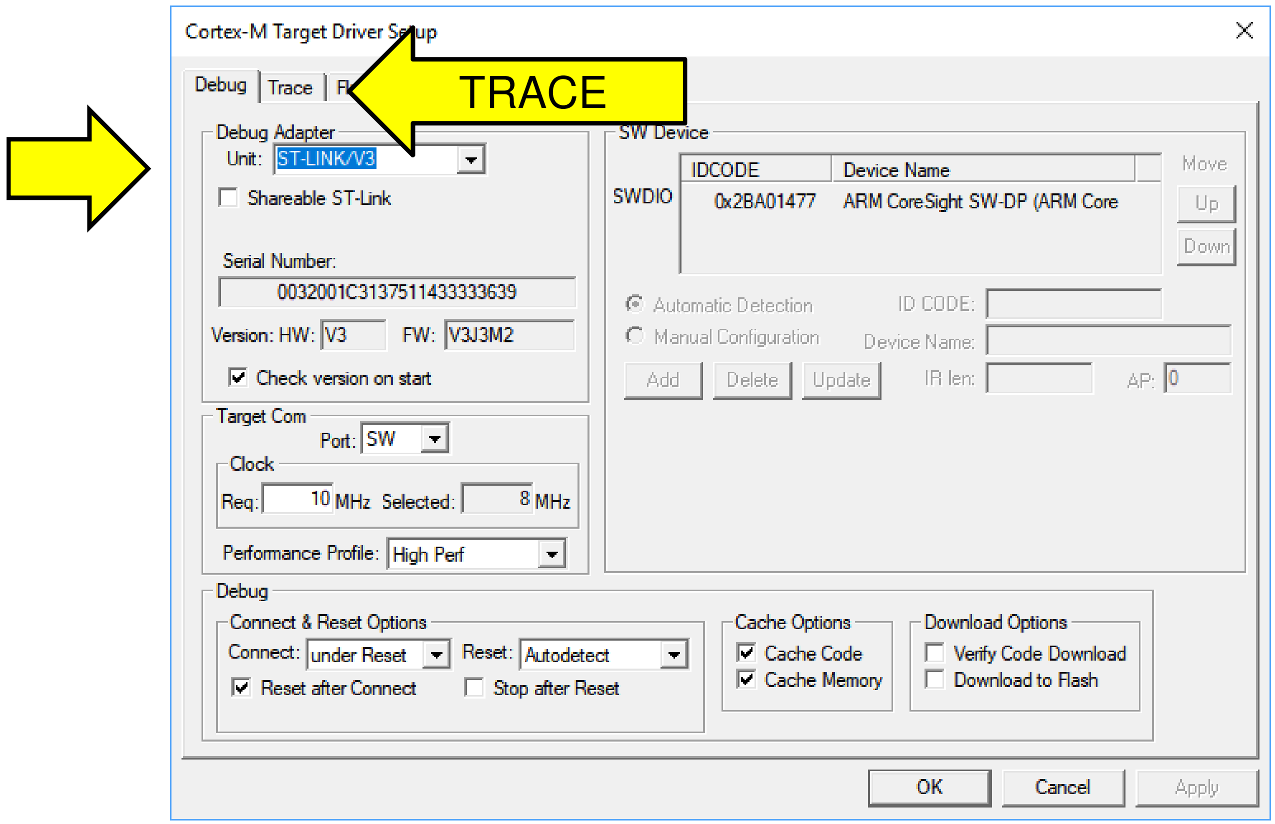
Clock speed and Debug



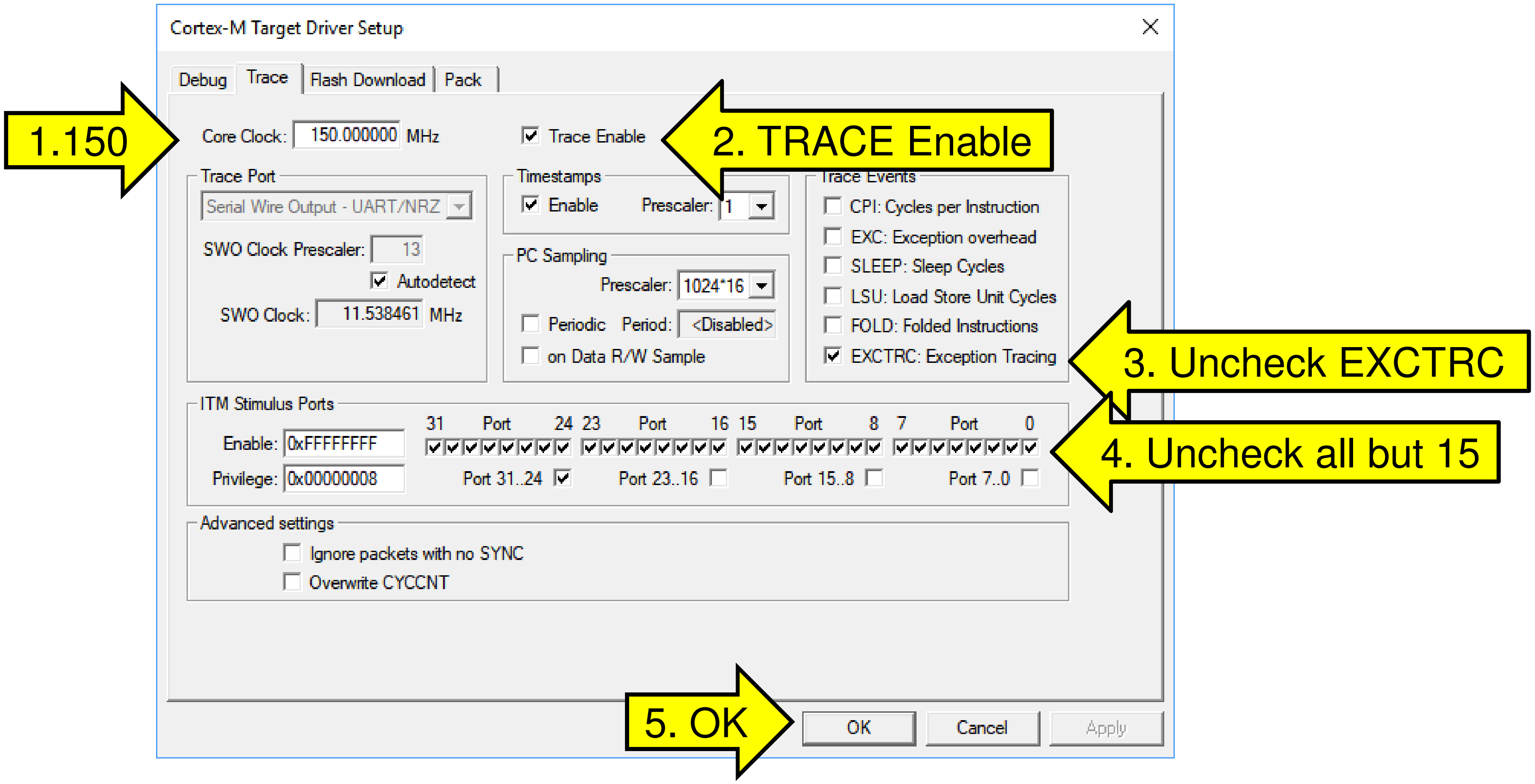
ST-Link Debugger - Settings



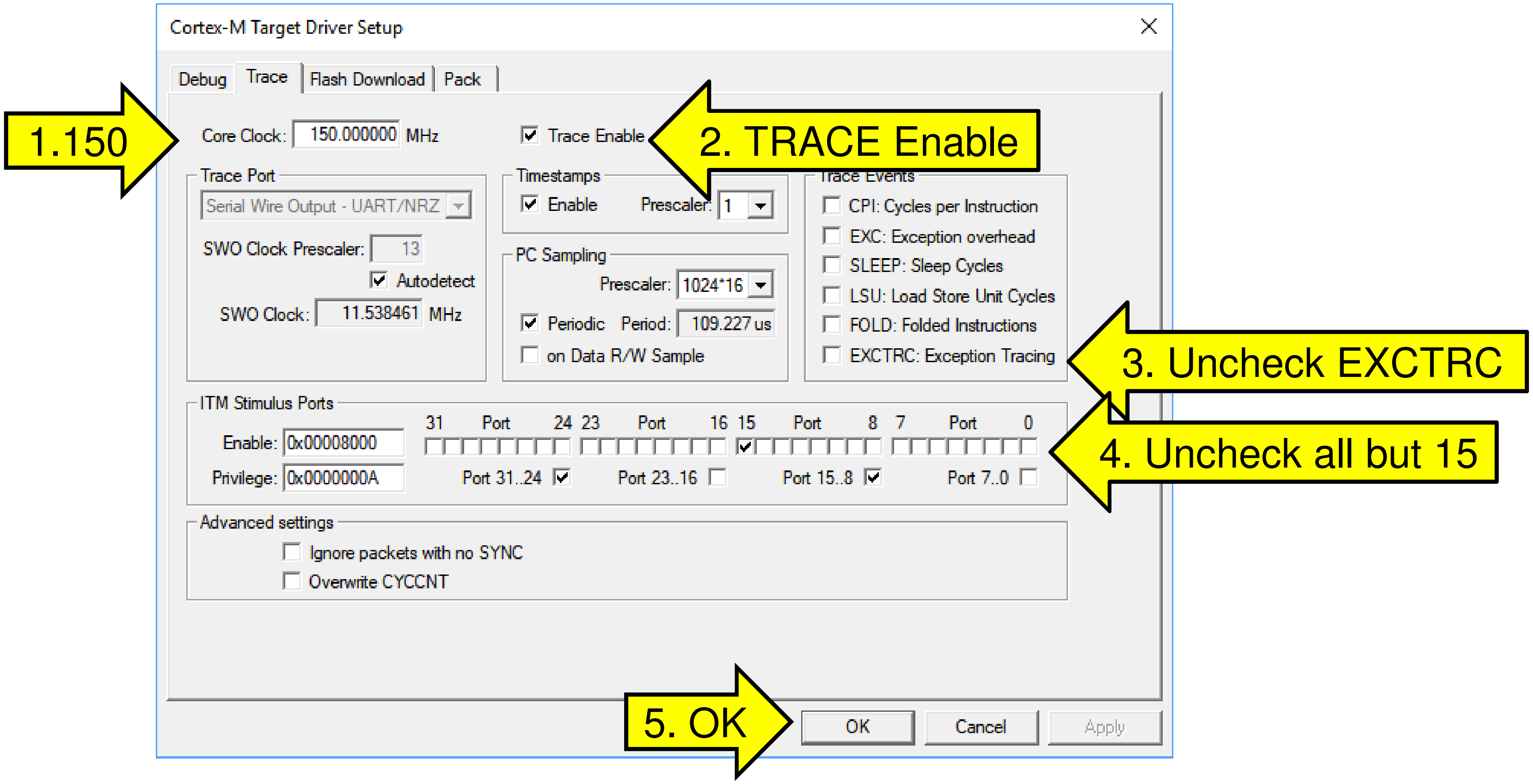
ST-Link/V3 & TRACE



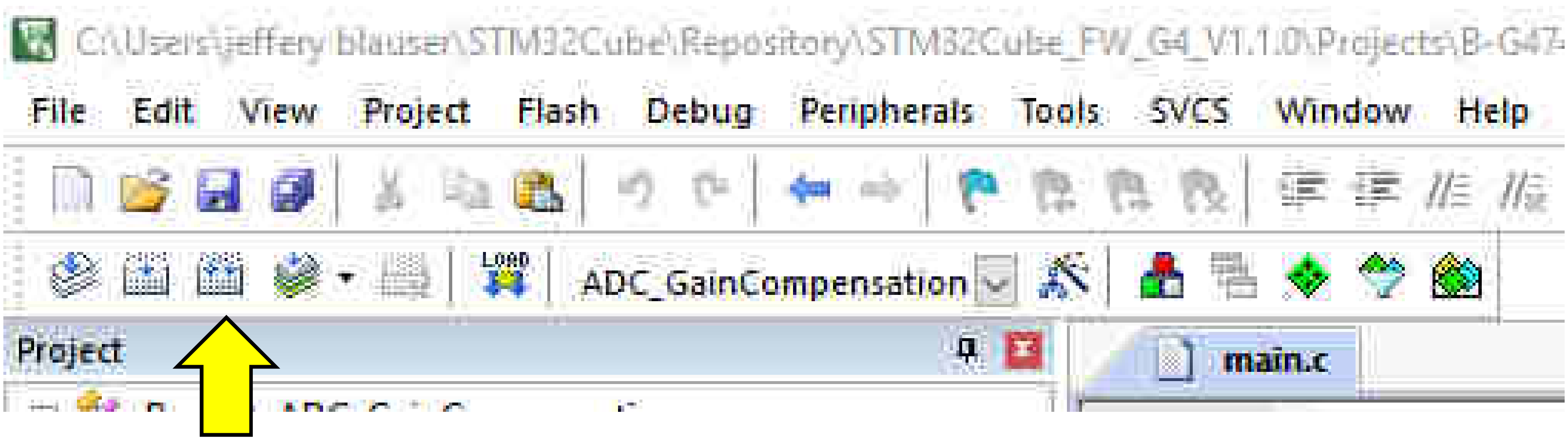
Core Clock & TRACE Enable (Before)



Core Clock & TRACE Enable (After)



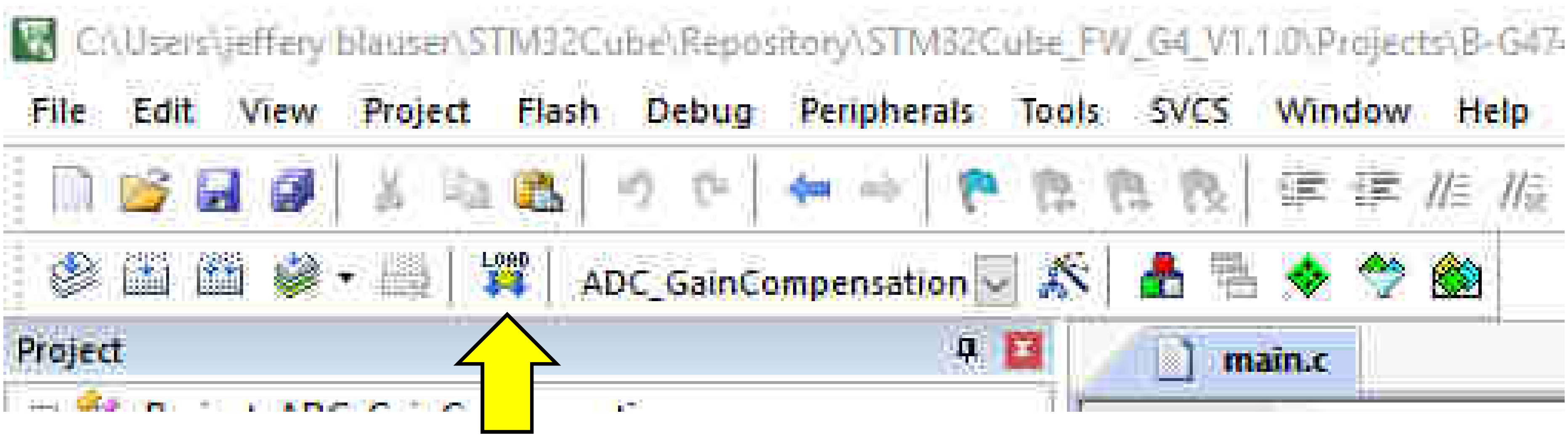
Build

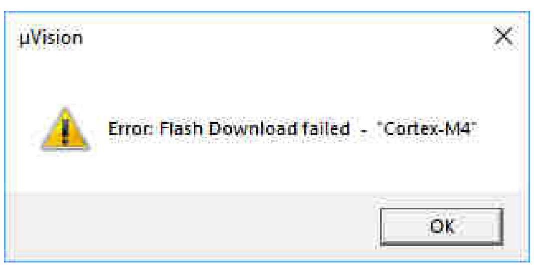


Build Output

```
Program Size: Code=6572 RO-data=576 RW-data=92 ZI-data=1204
FromELF: creating hex file...
DAC_DualConversionFromDMA\DAC_DualConversionFromDMA.hex: Warning: Q9931W: Your license for feature Keil will expire in 30 days
Finished: 0 information, 1 warning and 0 error messages.
"DAC_DualConversionFromDMA\DAC_DualConversionFromDMA.axf" - 0 Error(s), 0 Warning(s).
Build Time Elapsed:  00:01:38
```

Load



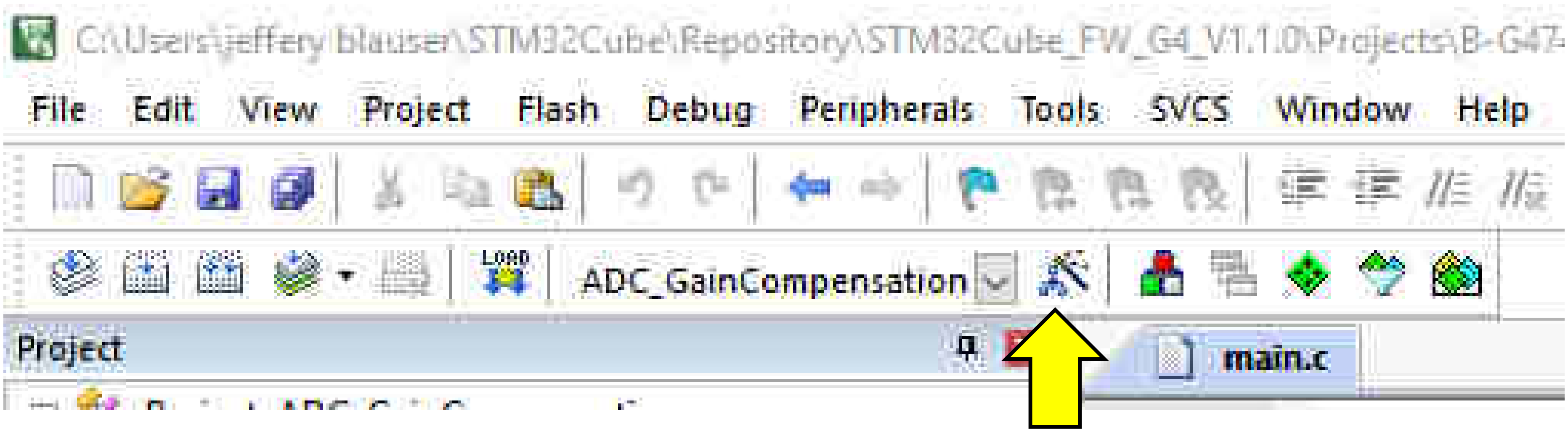


Build Output

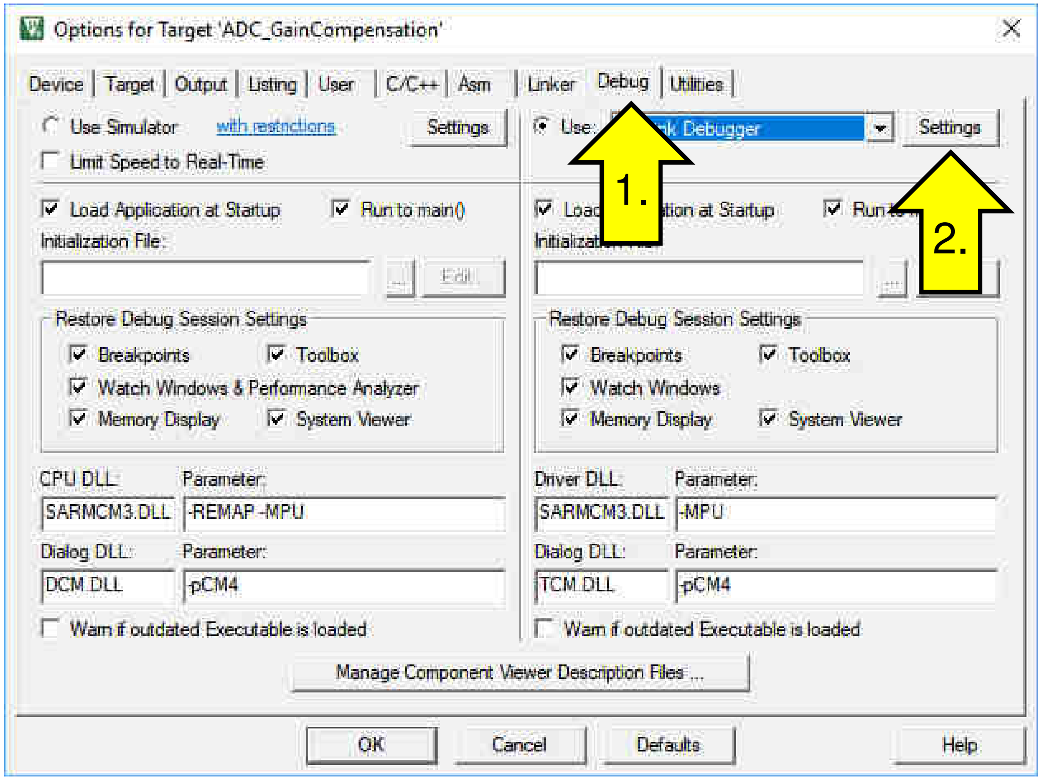
```
Contents mismatch at: 08001860H  (Flash=38H  Required=C5H) !
Contents mismatch at: 08001861H  (Flash=46H  Required=F8H) !
Contents mismatch at: 08001862H  (Flash=FEH  Required=88H) !
Contents mismatch at: 08001863H  (Flash=F7H  Required=00H) !
Contents mismatch at: 08001864H  (Flash=B9H  Required=60H) !
Contents mismatch at: 08001865H  (Flash=FCH  Required=6CH) !
Too many errors to display !
Error: Flash Download failed  -  "Cortex-M4"
```

Keil MDK-ARM

Options

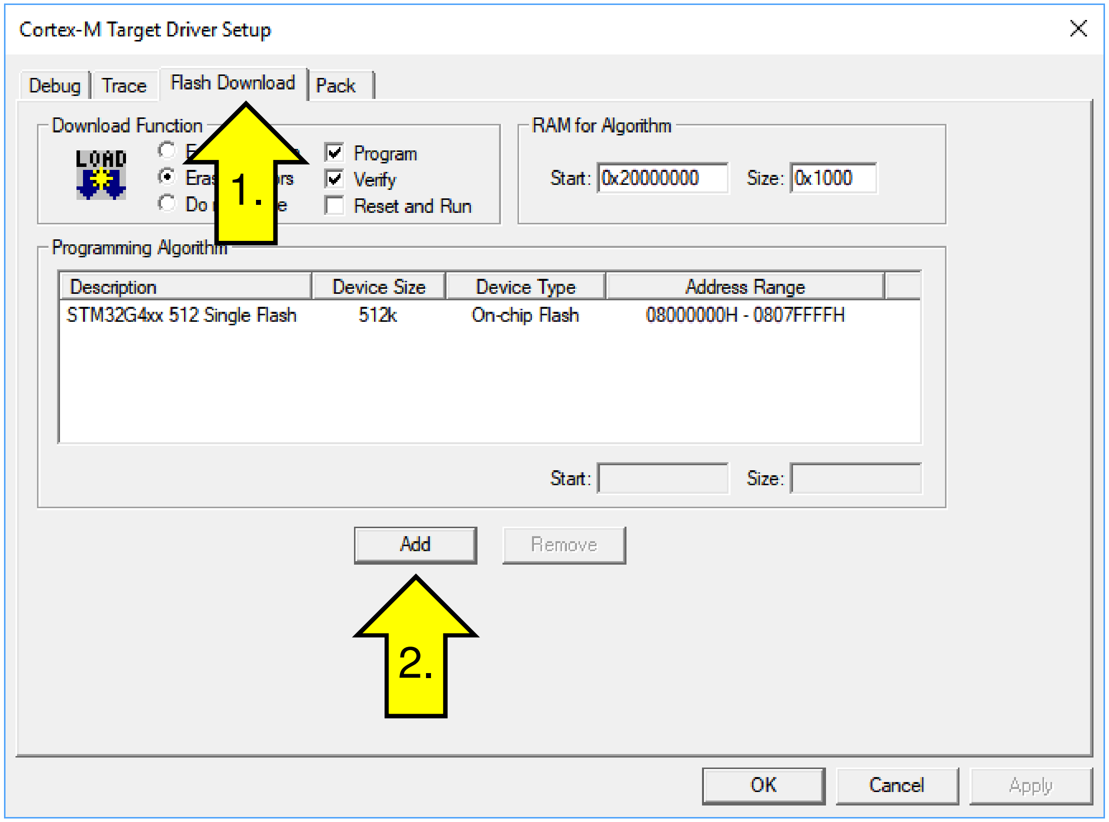


ST-Link Debugger - Settings

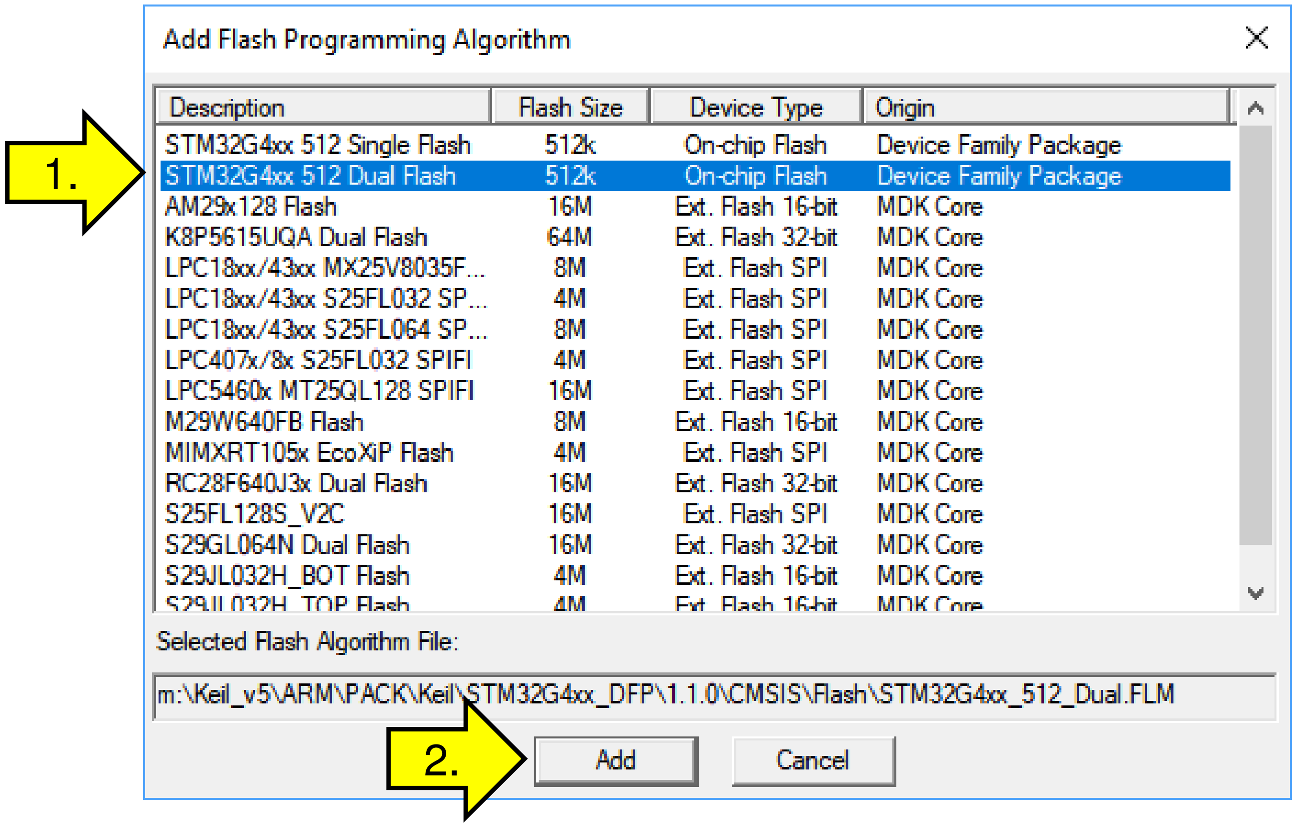


Keil MDK-ARM

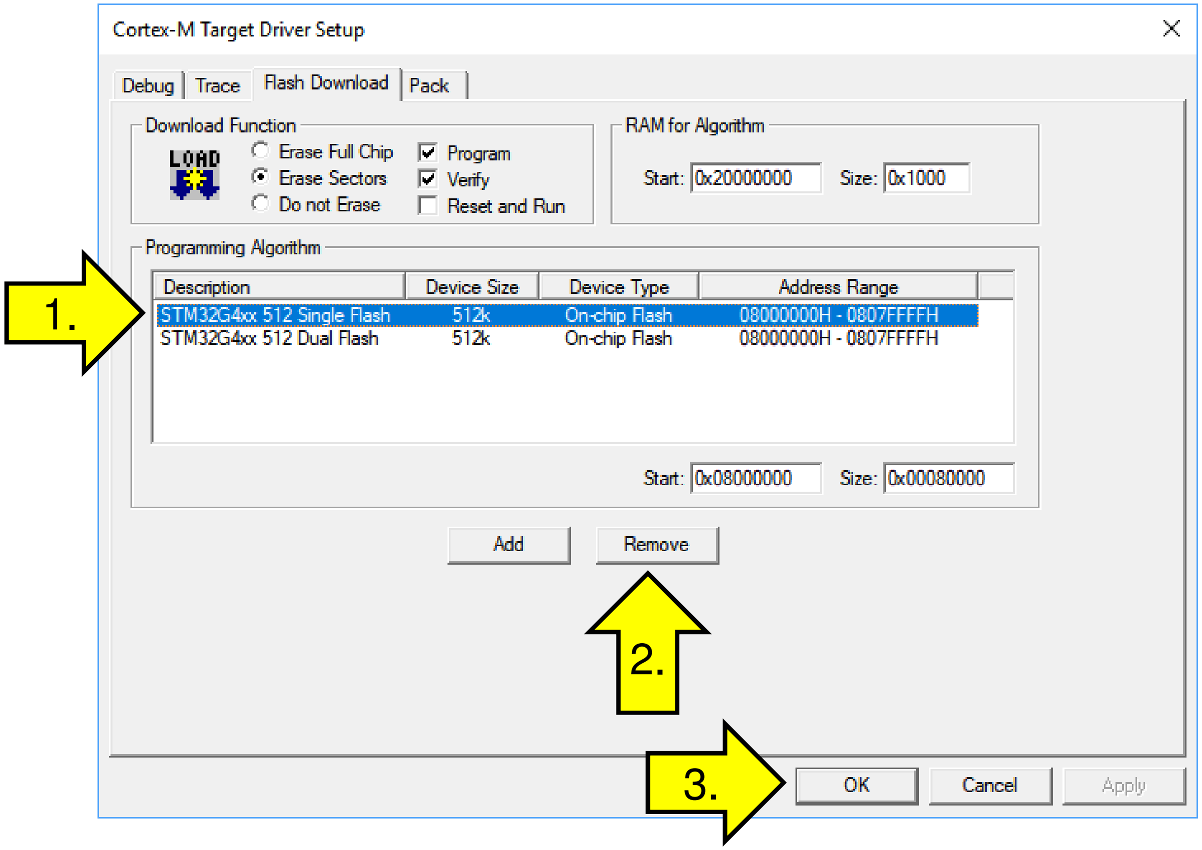
Flash Download → Add



STM32G4xx 512 Dual Flash



Remove Single Flash



Keil MDK-ARM

Place 2 new variables for the Analyzer – main.c about line 56

```
51  /* USER CODE BEGIN PV */
52  /* Private variables -------
53
54  uint32_t wave[4] = {0x07FF0
55
56  uint16_t Display_DAC1DOR1;
57  uint16_t Display_DAC1DOR2;
58
59  /* USER CODE END PV */
```

See c:\2019_STM32G4_Workshop\HandsOn_Labs\DAC
For the file CodeLinesForMain_DAC.txt for cut and paste.

Keil MDK-ARM

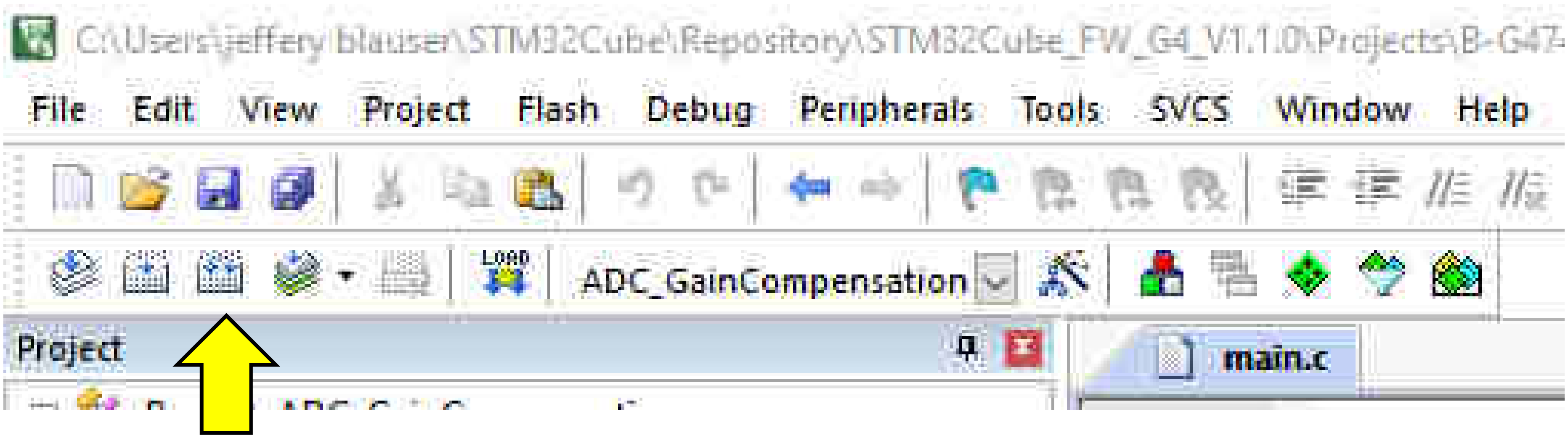
Execute 3 new lines for the Analyzer – main.c about line 143

```
138   while (1)
139   {
140     /* USER CODE END WHILE */
141
142     /* USER CODE BEGIN 3 */
143     Display_DAC1DOR1 = DAC1->DOR1;
144     Display_DAC1DOR2 = DAC1->DOR2;
145     HAL_Delay(25);
146   }
```

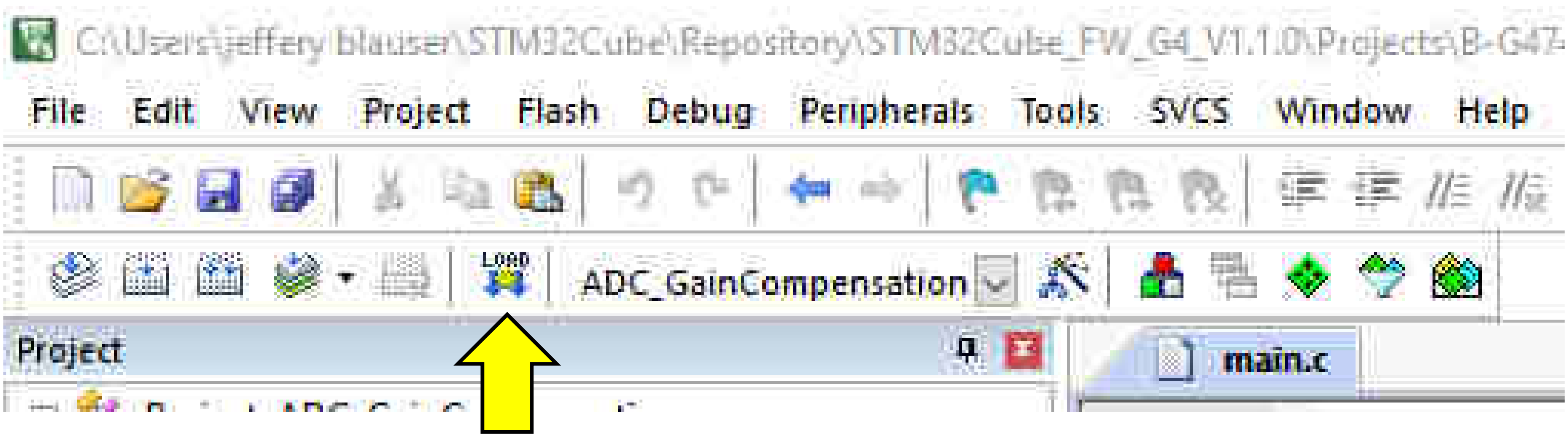
See c:\2019_STM32G4_Workshop\HandsOn_Labs\DAC
For the file CodeLinesForMain_DAC.txt for cut and paste.

Keil MDK-ARM

FILE→SAVE→Build



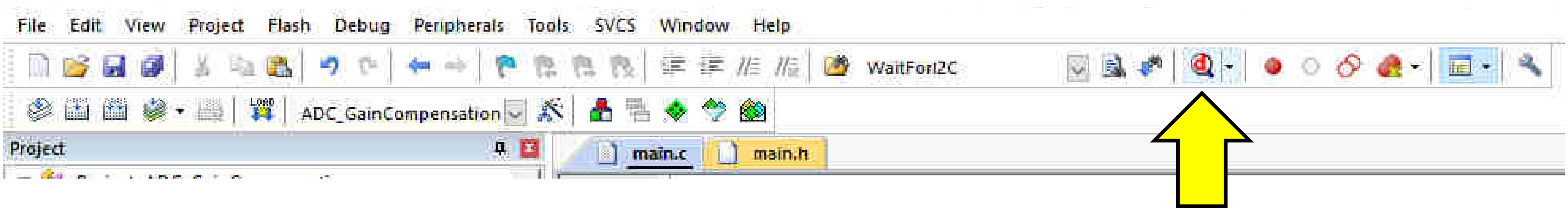
Load



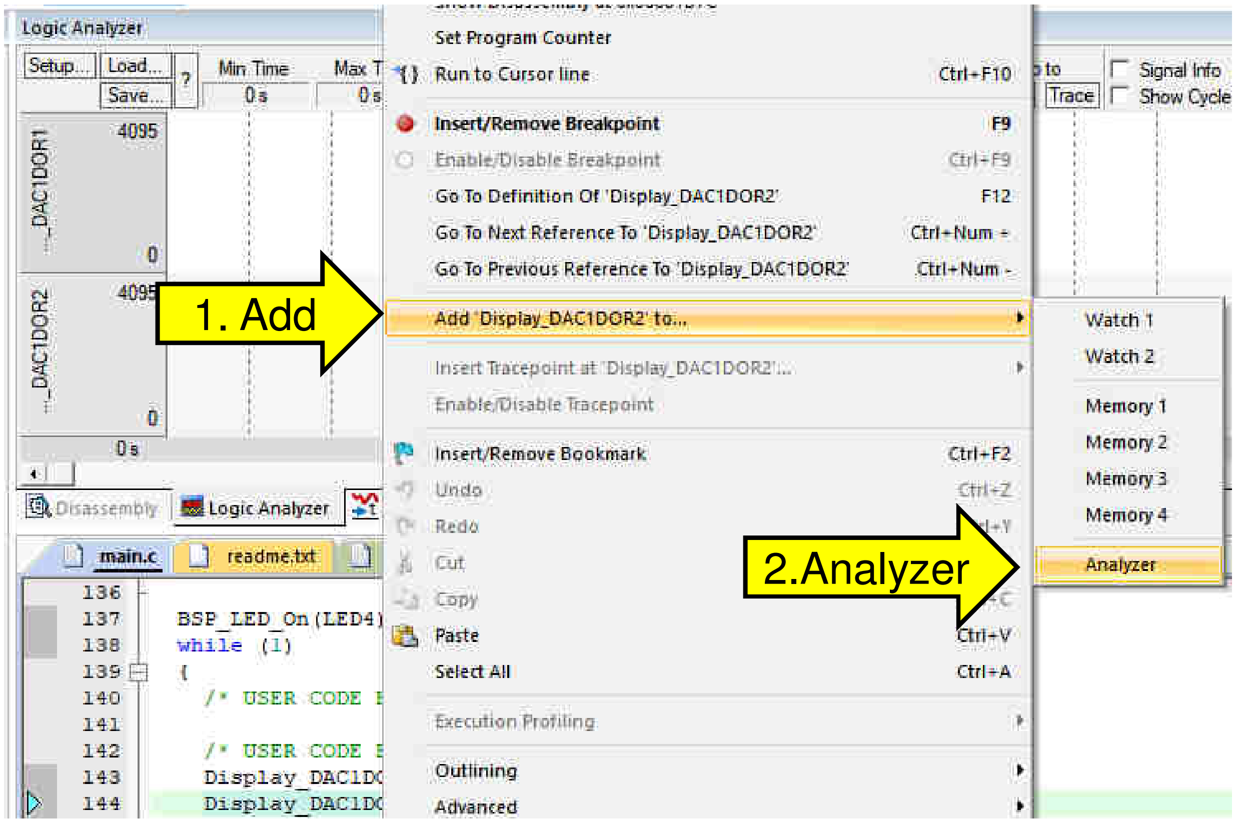
Build Output

```
Build Time Elapsed:  00:01:38
Load "DAC_DualConversionFromDMA\\DAC_DualConversionFromDMA.axf"
Erase Done.
Programming Done.
Verify OK.
Flash Load finished at 14:51:03
```

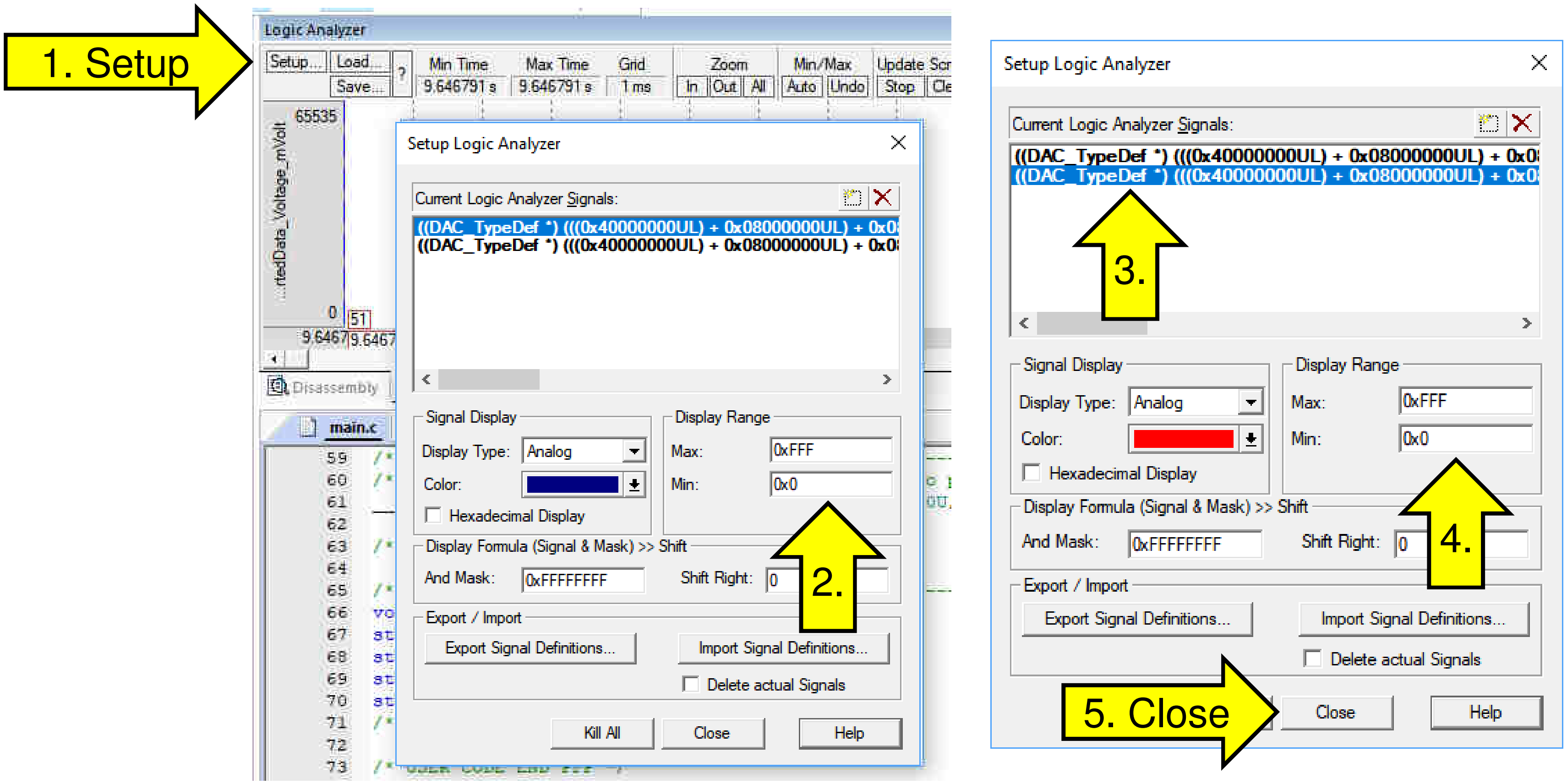
Debugger


File
Edit
View
Project
Flash
Debug
Peripherals
Tools
SVCS
Window
Help
WaitForI2C
ADC_GainCompensation
Project
main.c
main.h

Add to Analyzer Display_DAC1DOR1 & 2



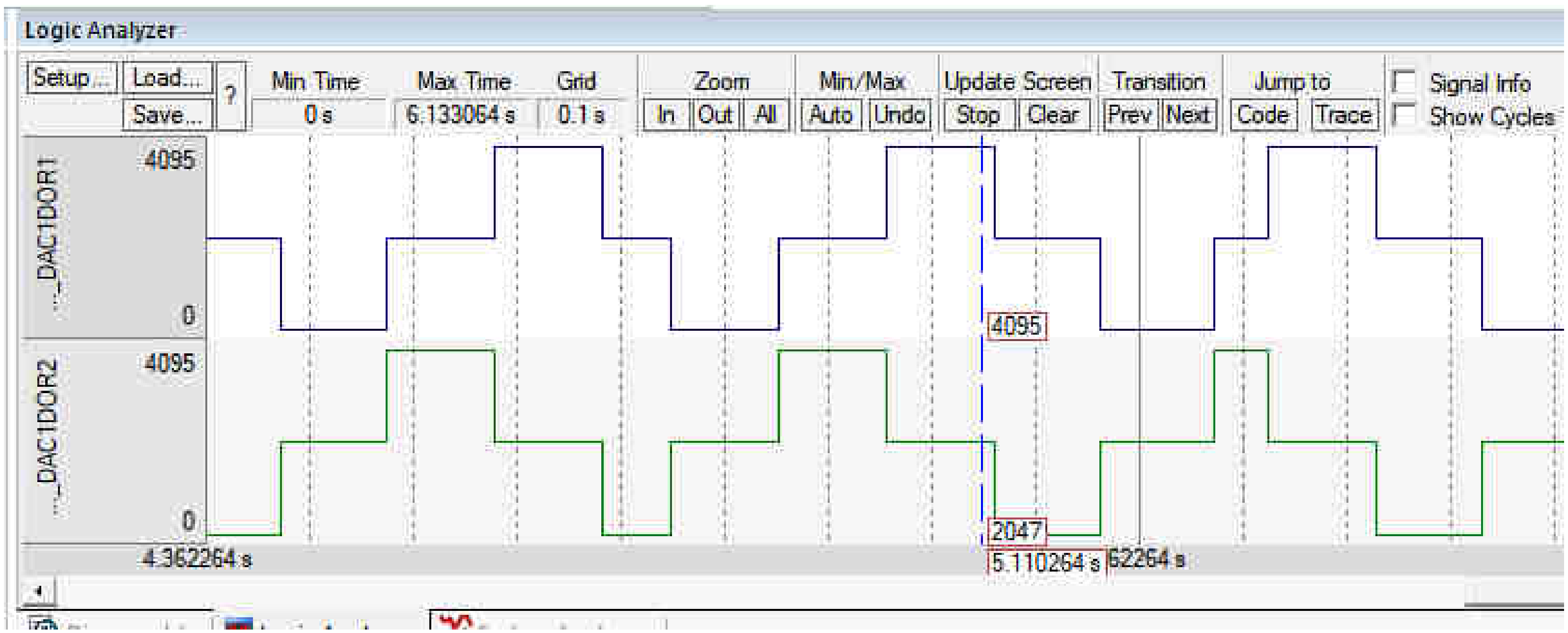
Setup – Min=0x0, Max = 0xFFF



Keil MDK-ARM

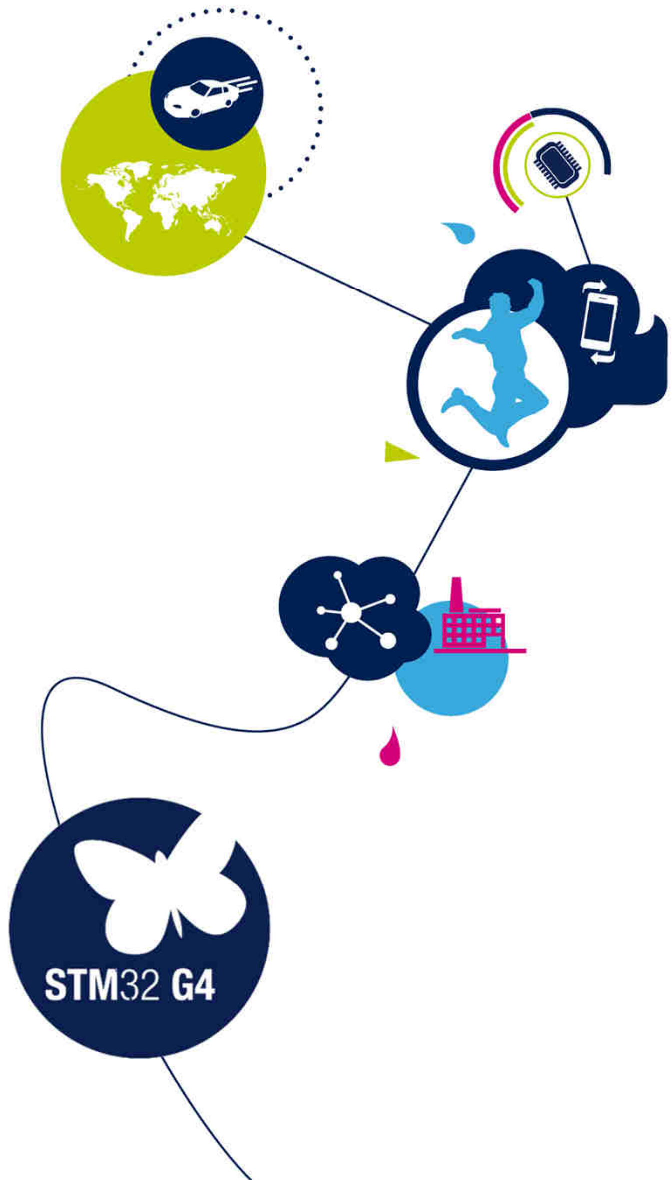
RUN

Logic Analyzer Output



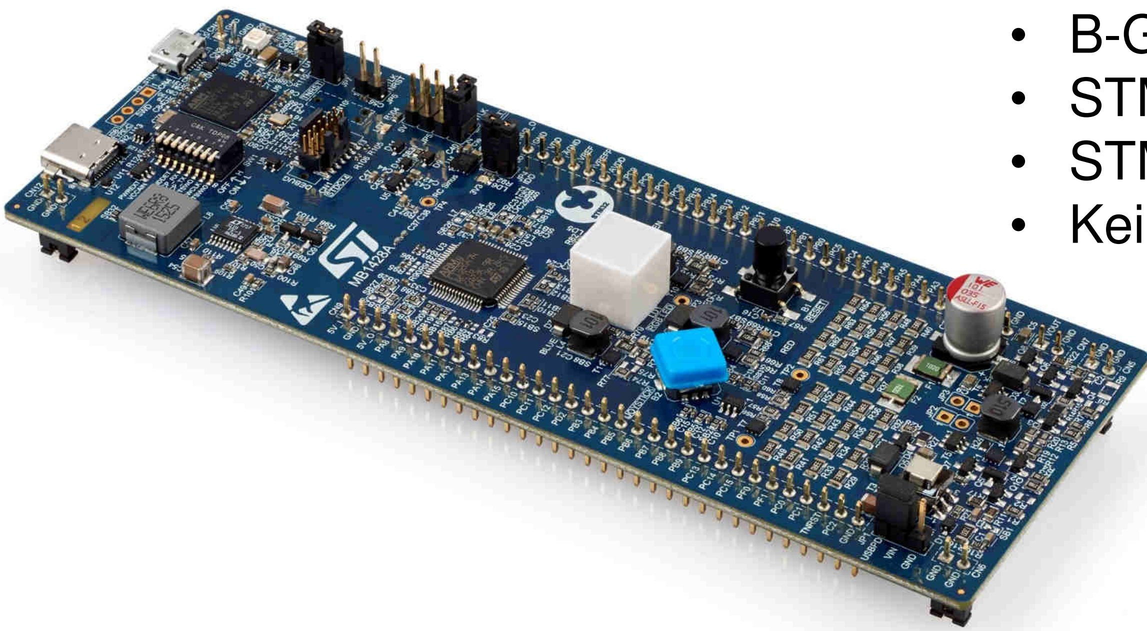
STM32G4 mixed-signal MCU workshop

Hands On:
Programmable Gain Amplifier



life.augmented

Systems Check



- USB cable
- B-G474E-DPOW1 Discovery Board
- STM32G4 Family Cube Library
- STM32CubeMx
- Keil MDK-ARM

Programmable Gain Amplifiers

G4 PGA

- 6 PGA's available
- Non inverting gain 2/4/8/16/32/64
- Inverting gain -1/-3/-7/-15/-31/-63
- Rail-to-Rail input and output
- Low input bias current
- Low input offset voltage
- 15MHz GBW product
- High speed mode for better slew rate

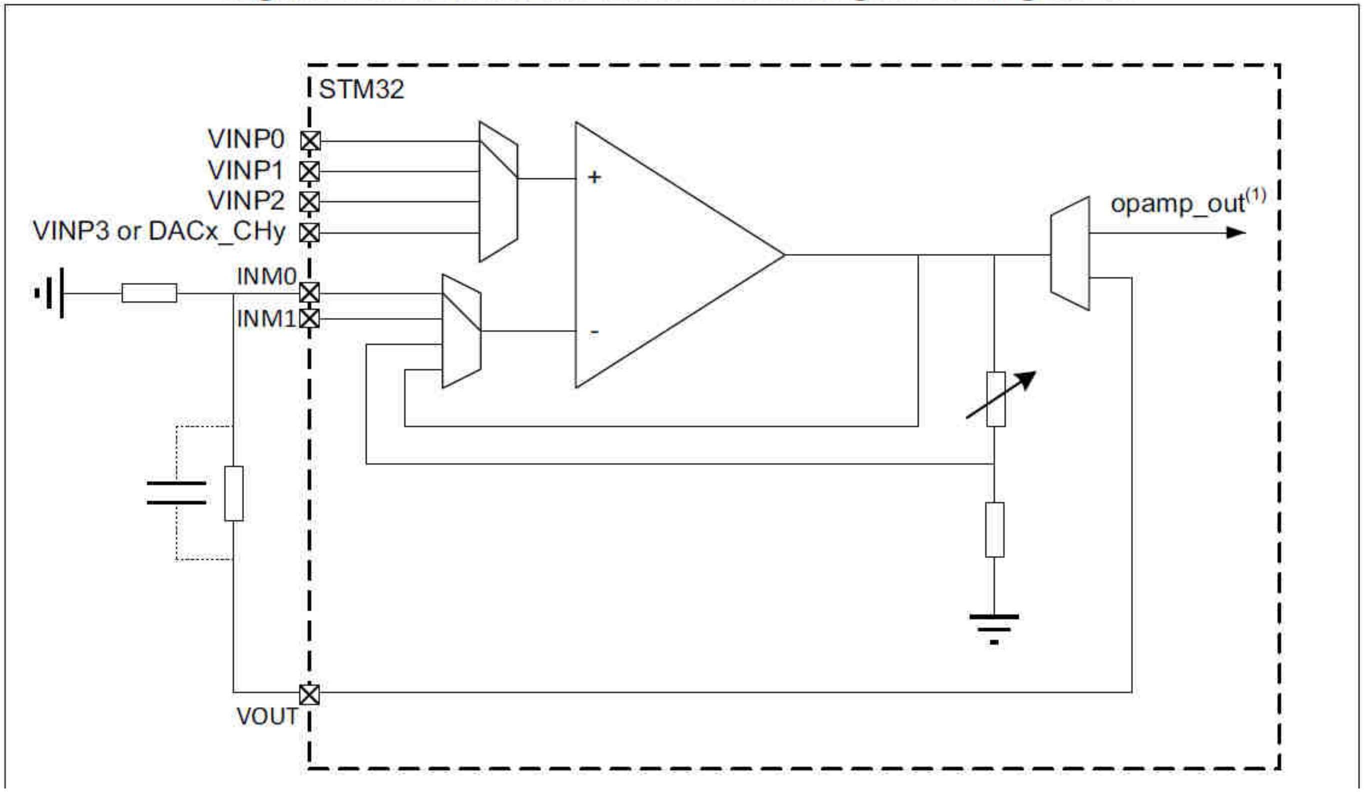
Internally connected to:

- ADC inputs
- External pins, both inputs and outputs
- Internal resistors
- DAC3 & DAC4 outputs

Programmable Gain Amplifiers

G4 Op Amp Stand Alone Mode

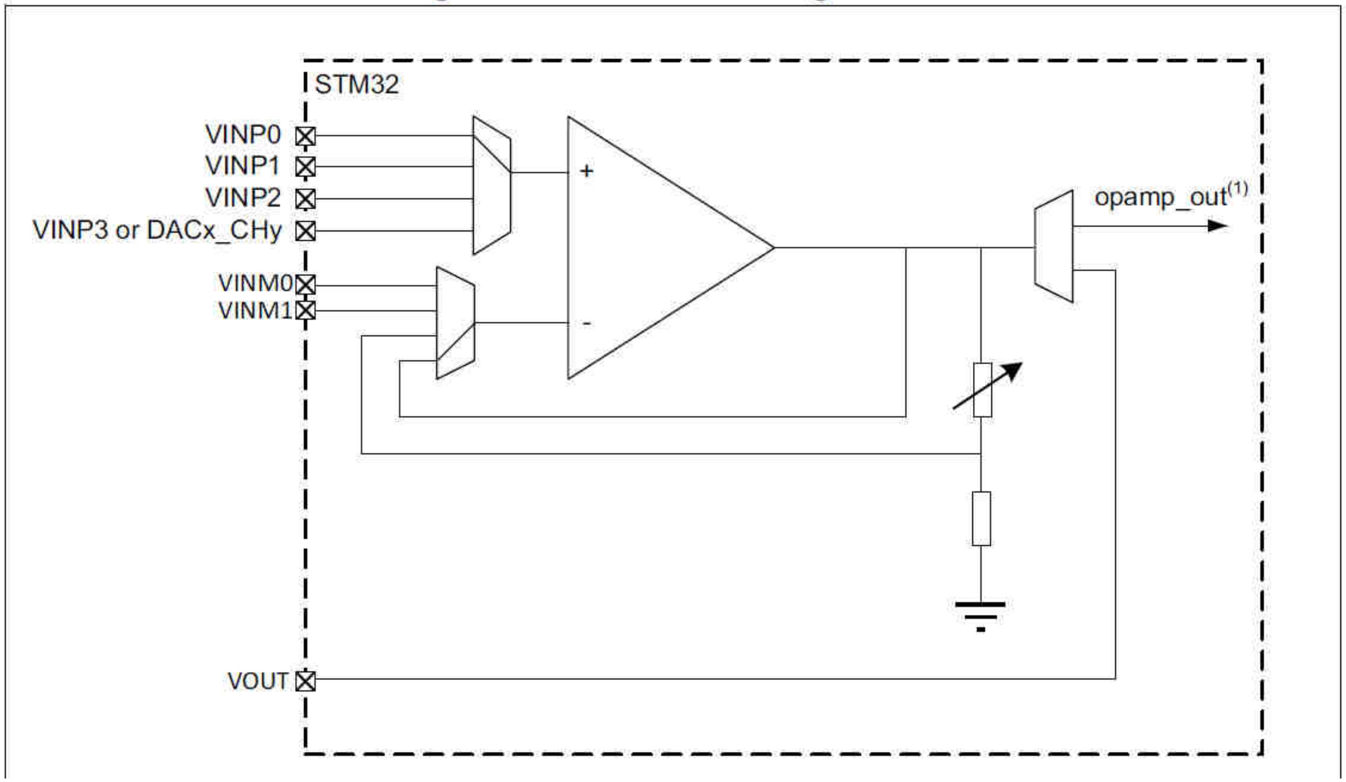
Figure 168. Standalone mode: external gain setting mode



Programmable Gain Amplifiers

G4 Op Amp Follower Mode

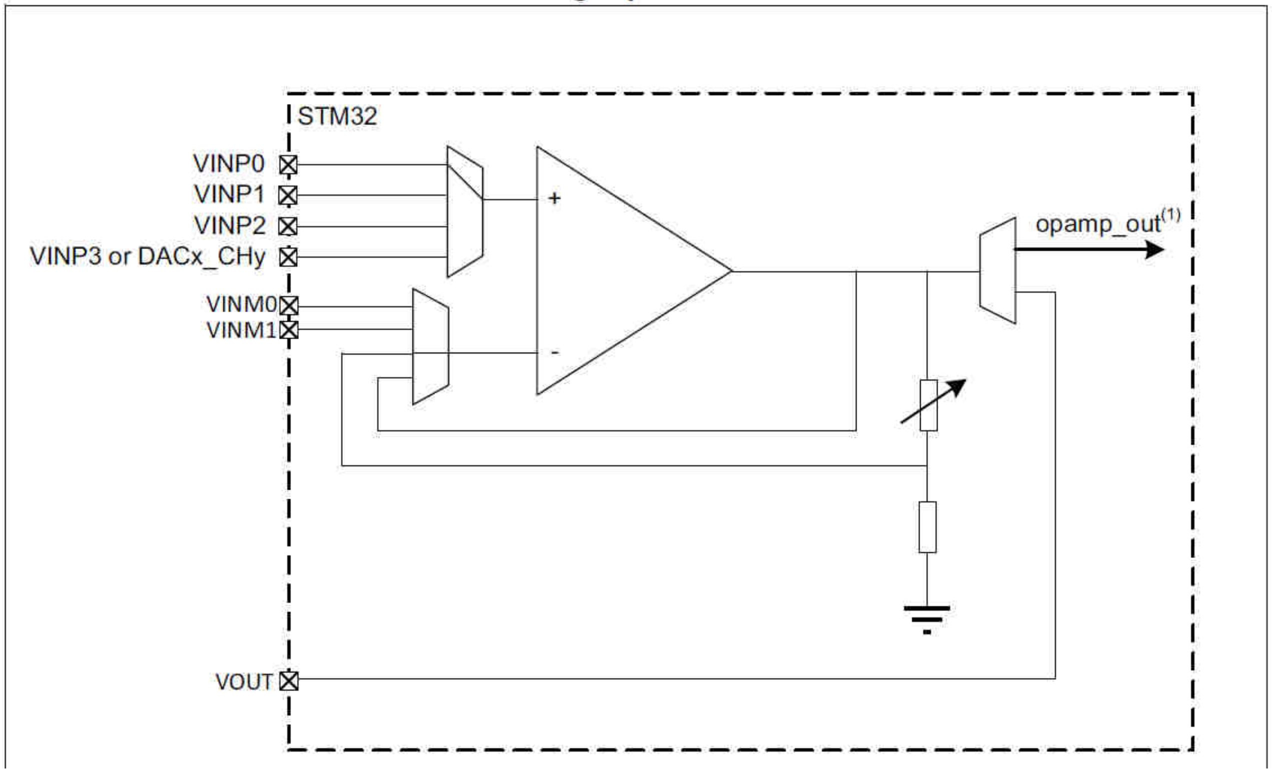
Figure 169. Follower configuration



Programmable Gain Amplifiers

G4 Op Amp PGA Mode

Figure 170. PGA mode, internal gain setting (x2/x4/x8/x16/x32/x64), inverting input not used



PGA Hands On

Lab steps 1 of 2

- The DMA provides samples (sinewave) to the DAC.
    - The DAC peripheral generates a sine wave signal on DAC1_OUT1(PA4) which is amplified by the OPAMP3.
    - The OPAMP3 amplified output is on PB1 with gain of either 2 or 4.
      The OPAMP gain is changed on the fly while the OPAMP remains enabled.
    - The test steps are:
    - Step 1:
        DAC: normal power mode
        OPAMP: normal power mode with gain = 2
        DMA: circular mode - DMA half transfer IT handled by CPU

life.augmented

PGA Hands On

Lab steps 2 of 2

- Step 2:

  DAC: normal power mode

  OPAMP: normal power mode

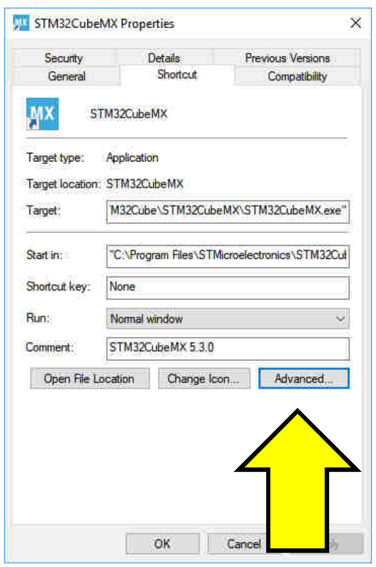
  - gain is changed from 2 to 4 or from 4 to 2

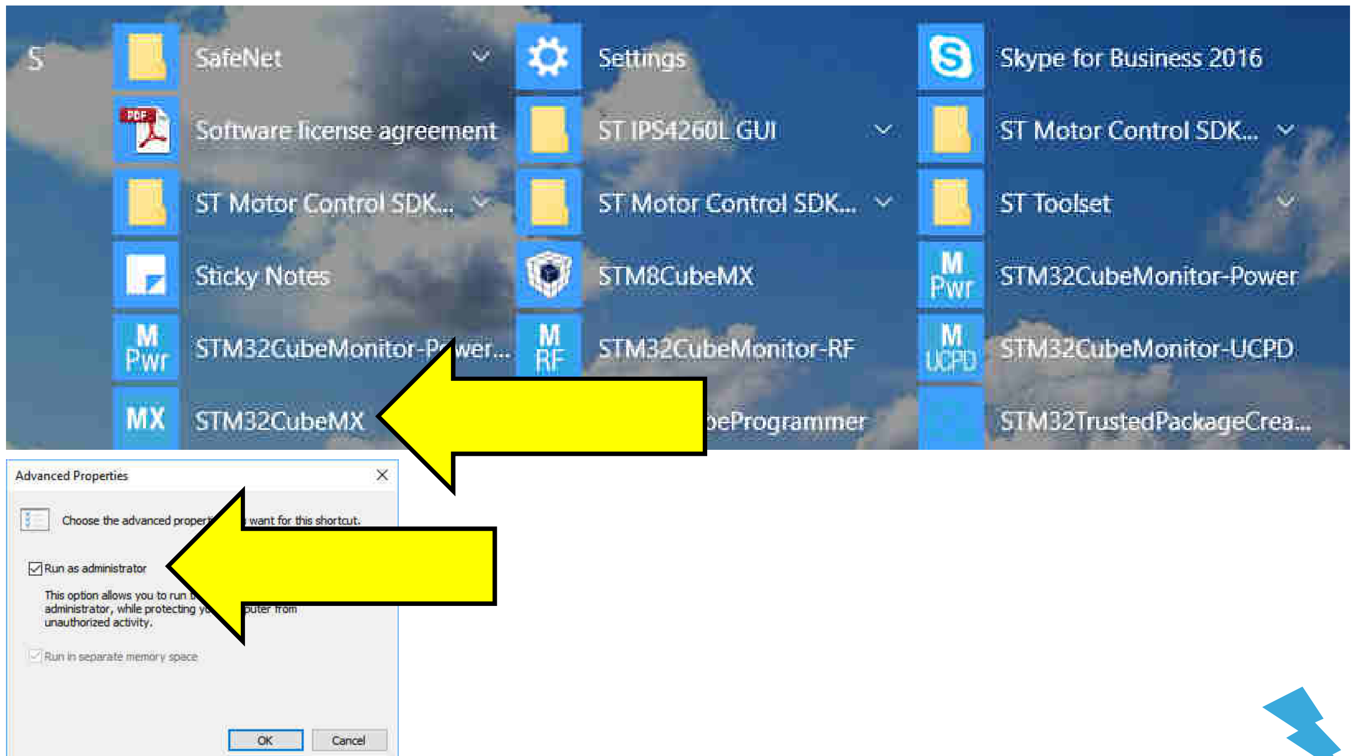
  DMA: circular mode - DMA half transfer IT handled by CPU

We will use the built in Logic Analyzer today instead of an o’scope on the DACs

STM32CubeMx

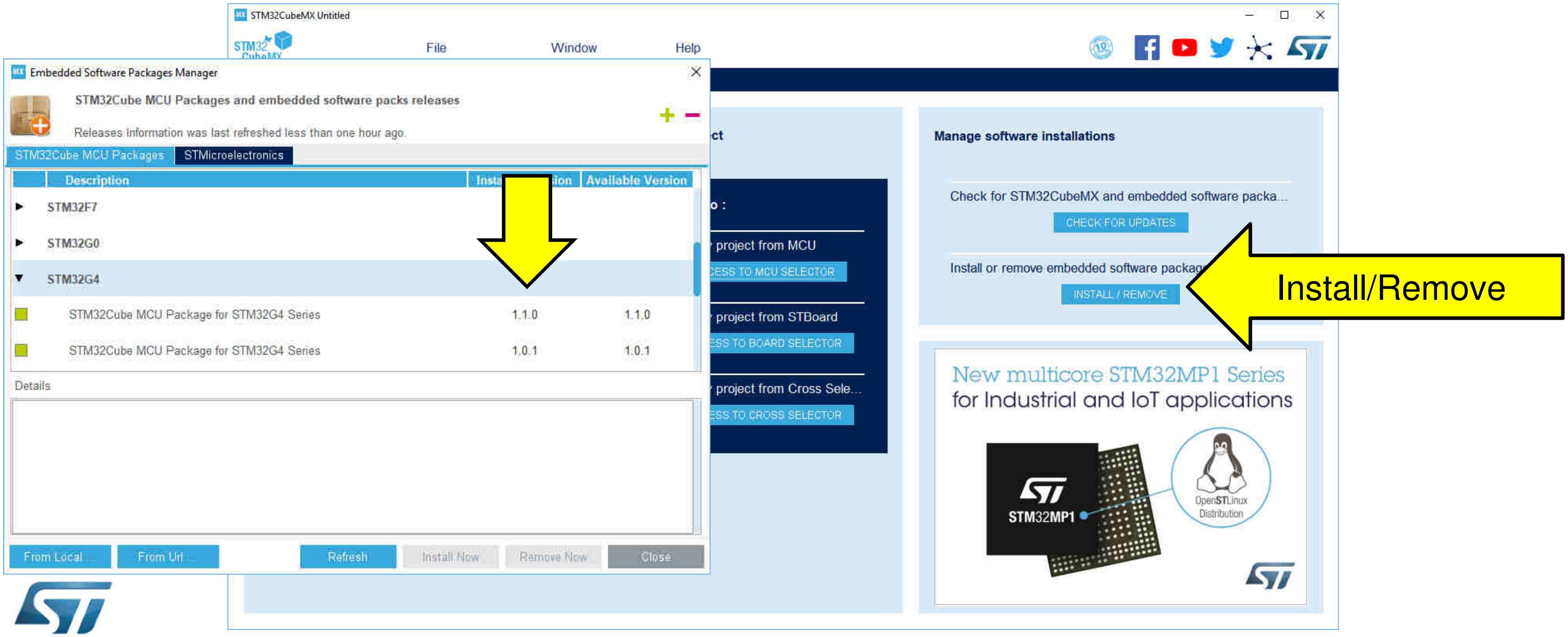
Start up





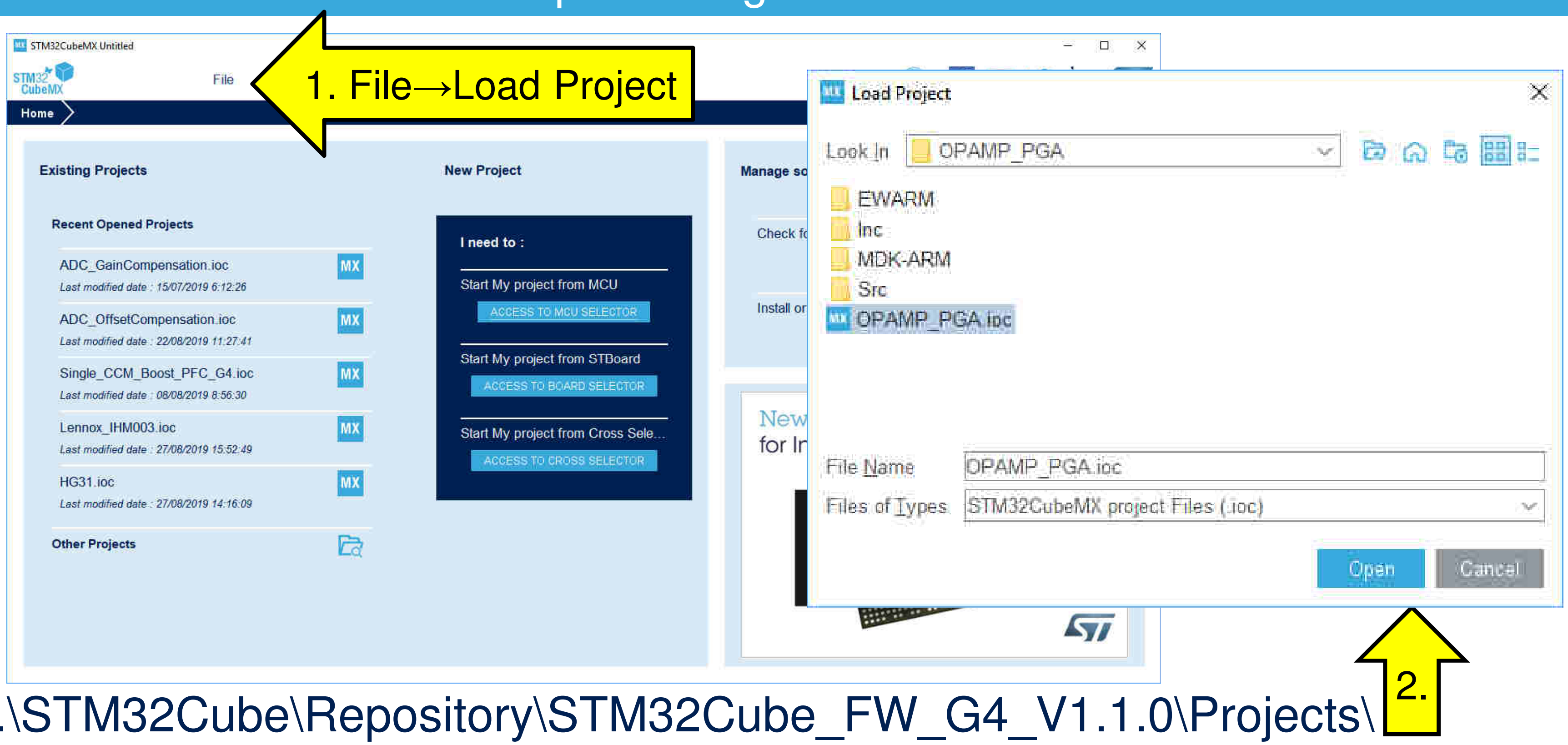
STM32CubeMx

Splash Page – G4 Library Installed?



life.augmented

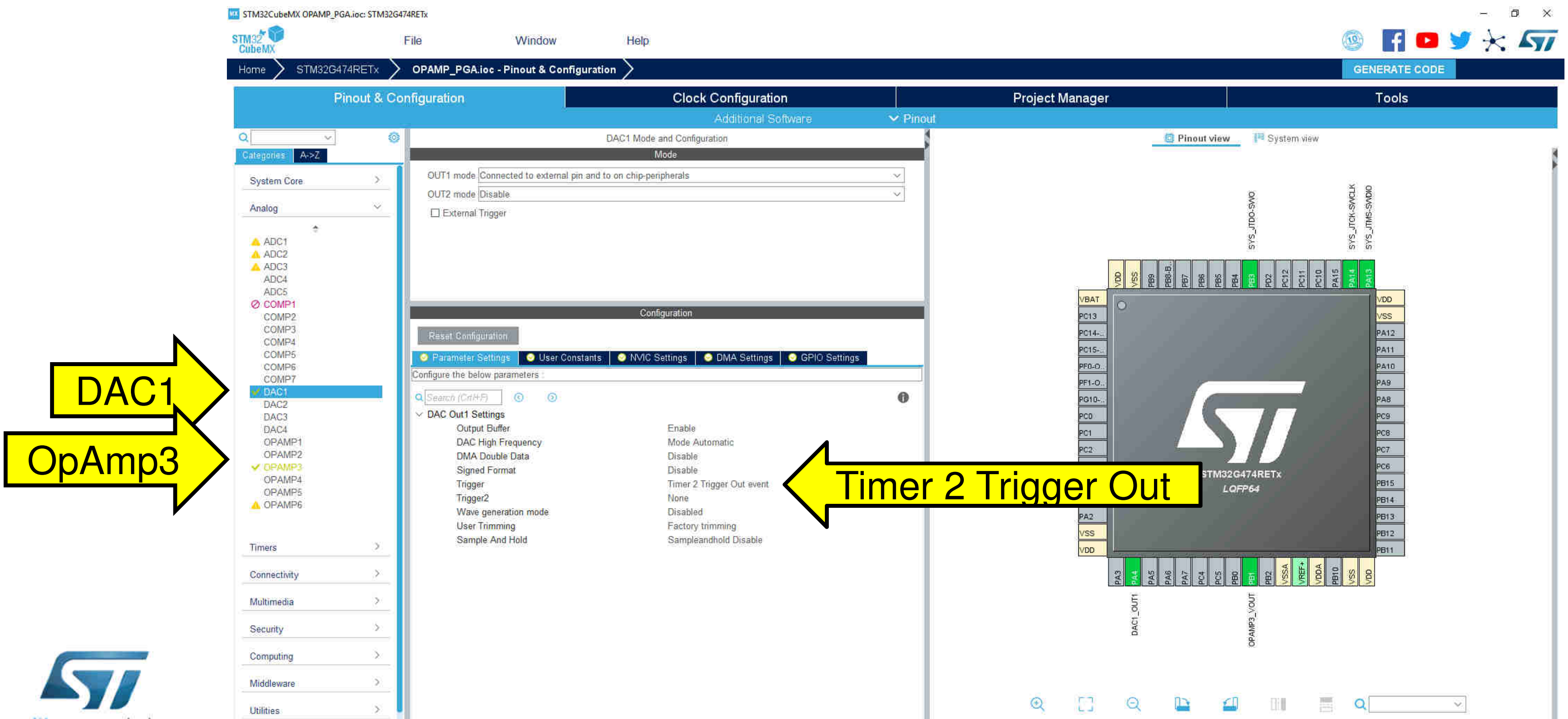
Splash Page



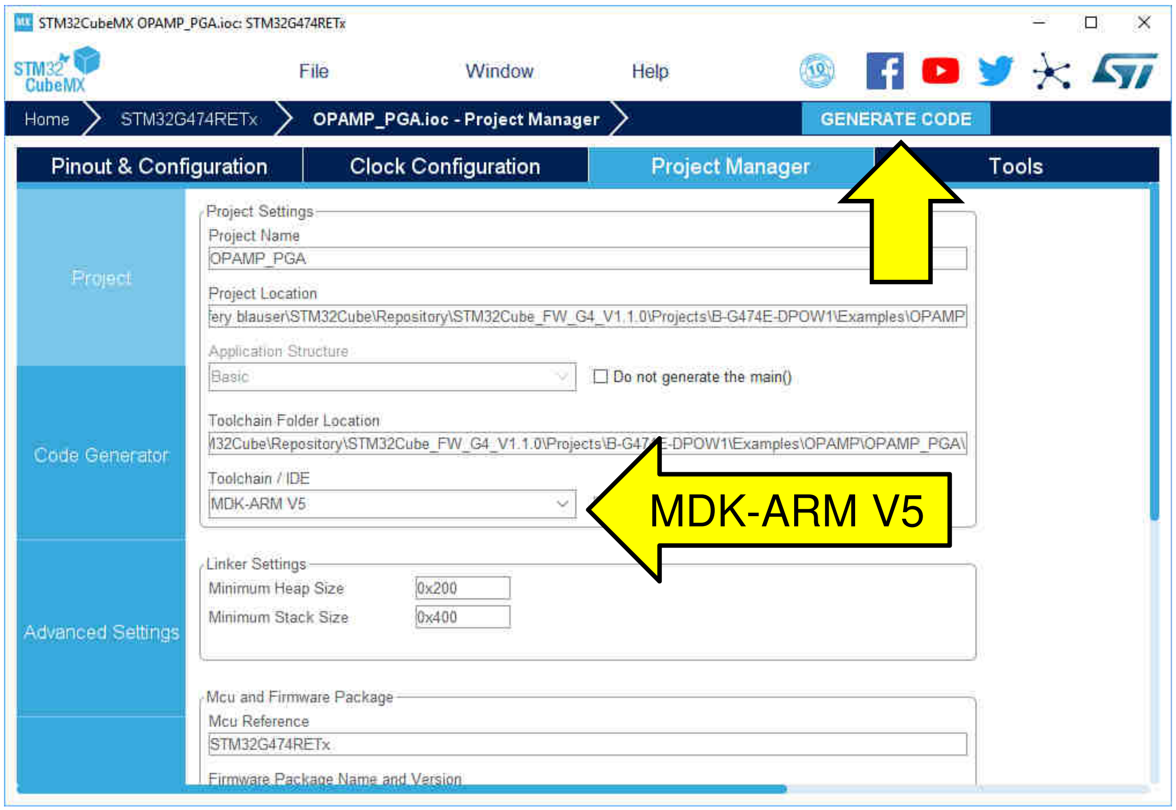
C:\...\STM32Cube\Repository\STM32Cube_FW_G4_V1.1.0\Projects\NUCLEO-G474RE\Examples\OPAMP\OPAMP_PGA

Graphic Design Representation – Click on Analog>

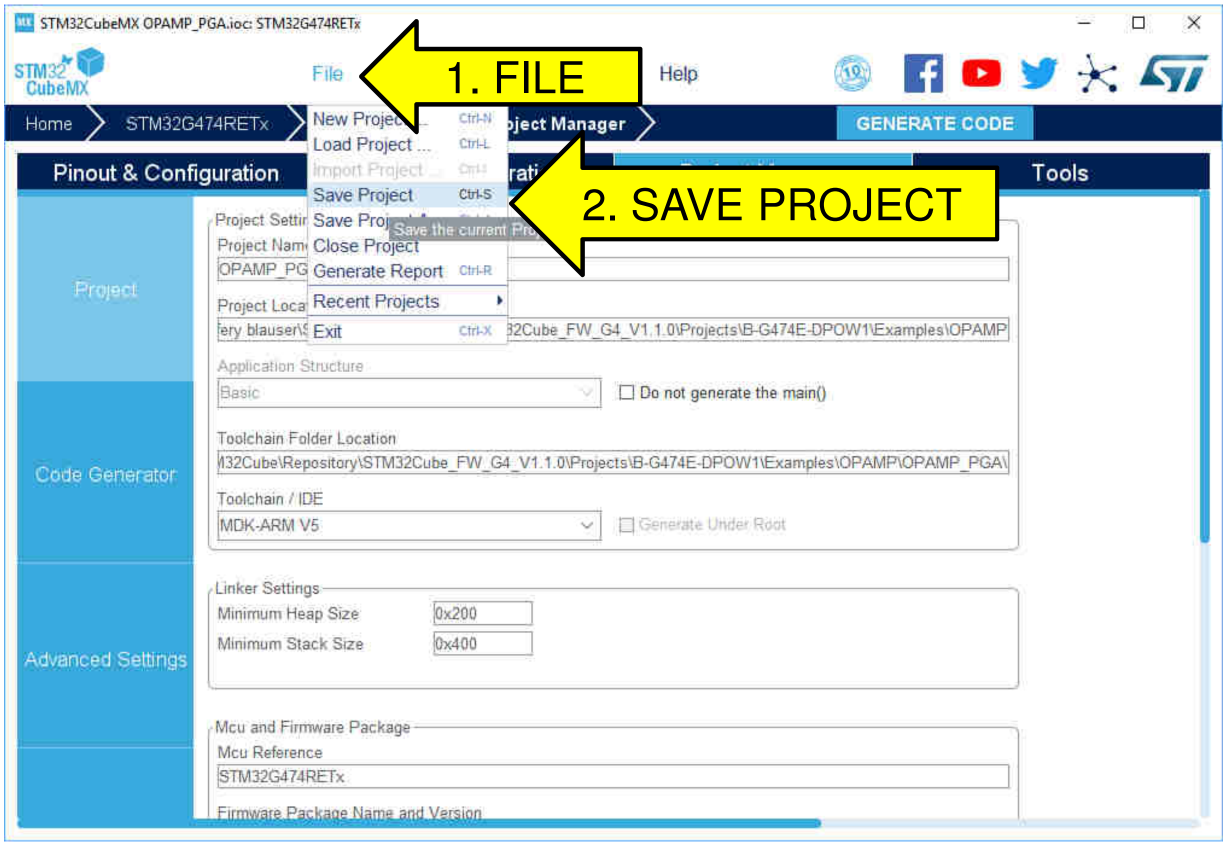
DAC1, Timer 2, Op Amp 3



Project Manager – Keil MDK-ARM V5



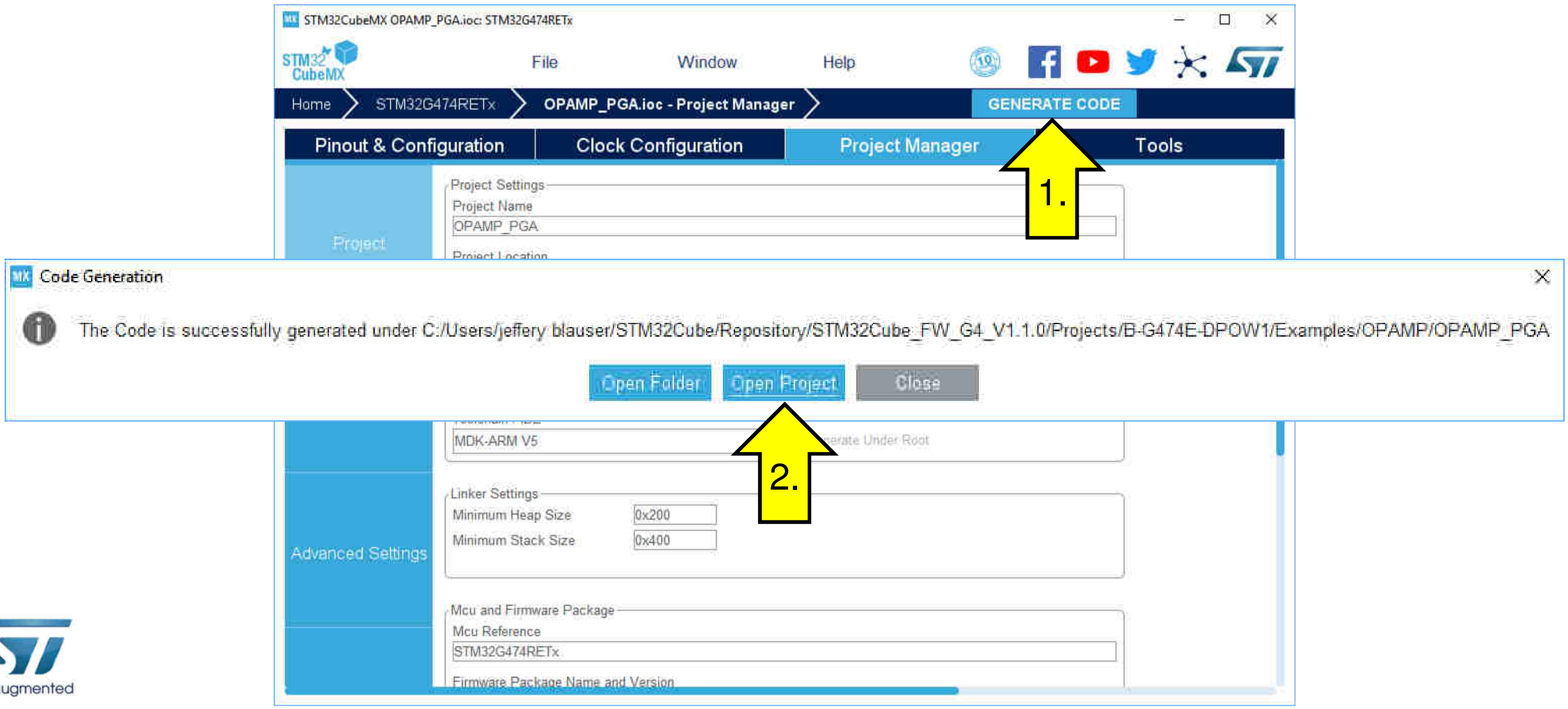
FILE→SAVE PROJECT



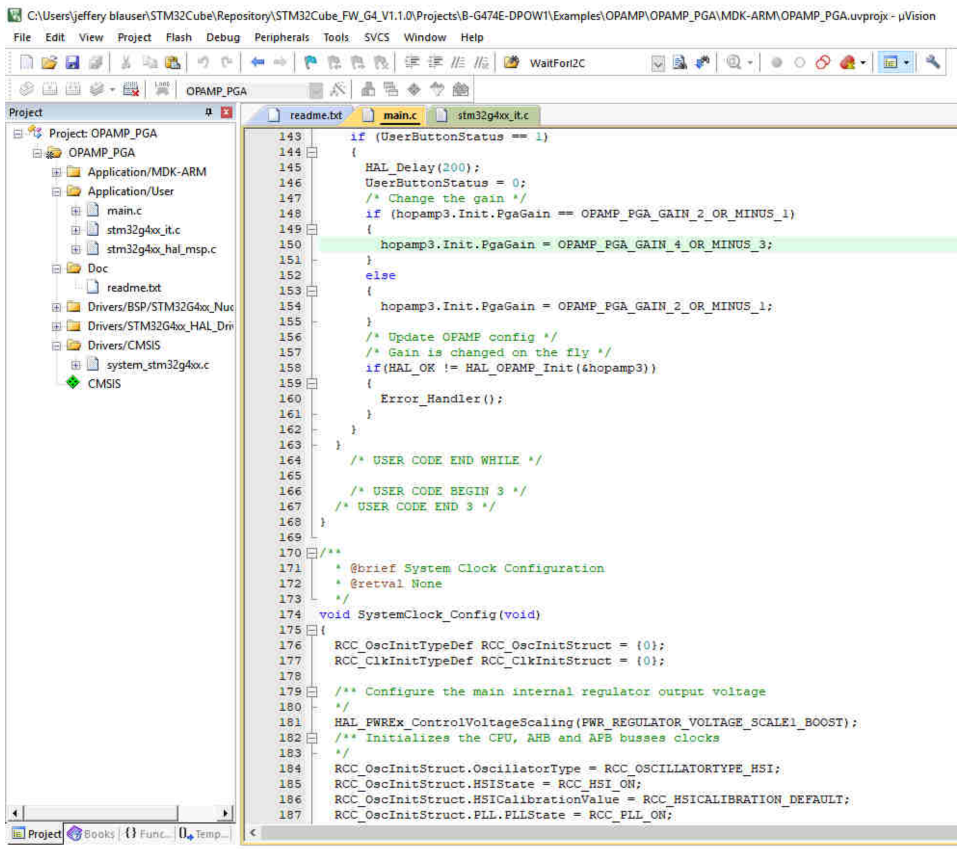
STM32 G4

life.augmented

Generate Code→Open Project

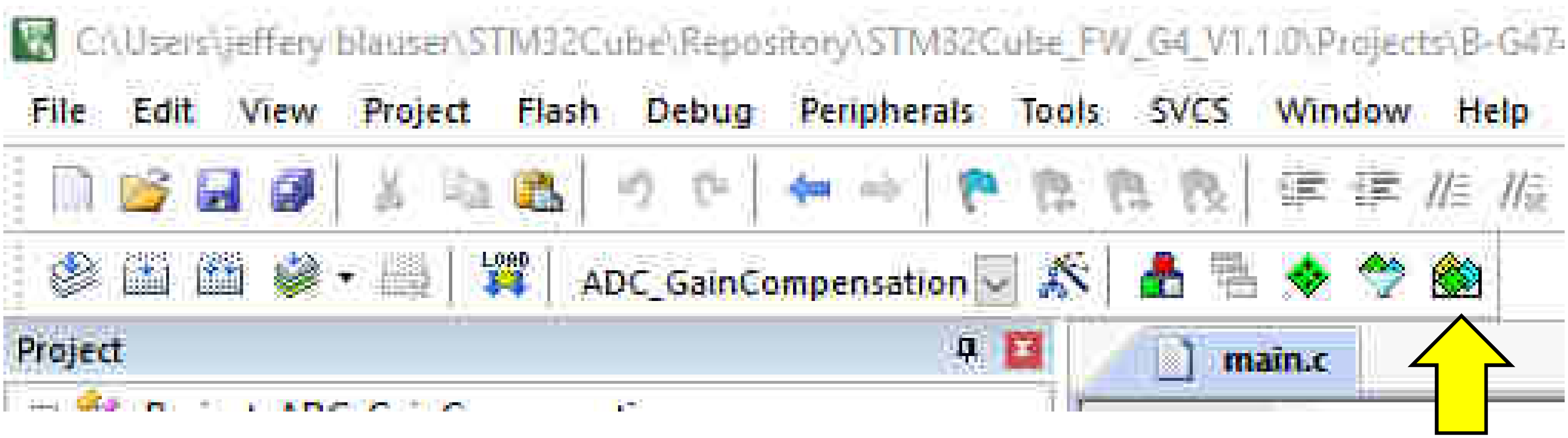


Keil Start up page



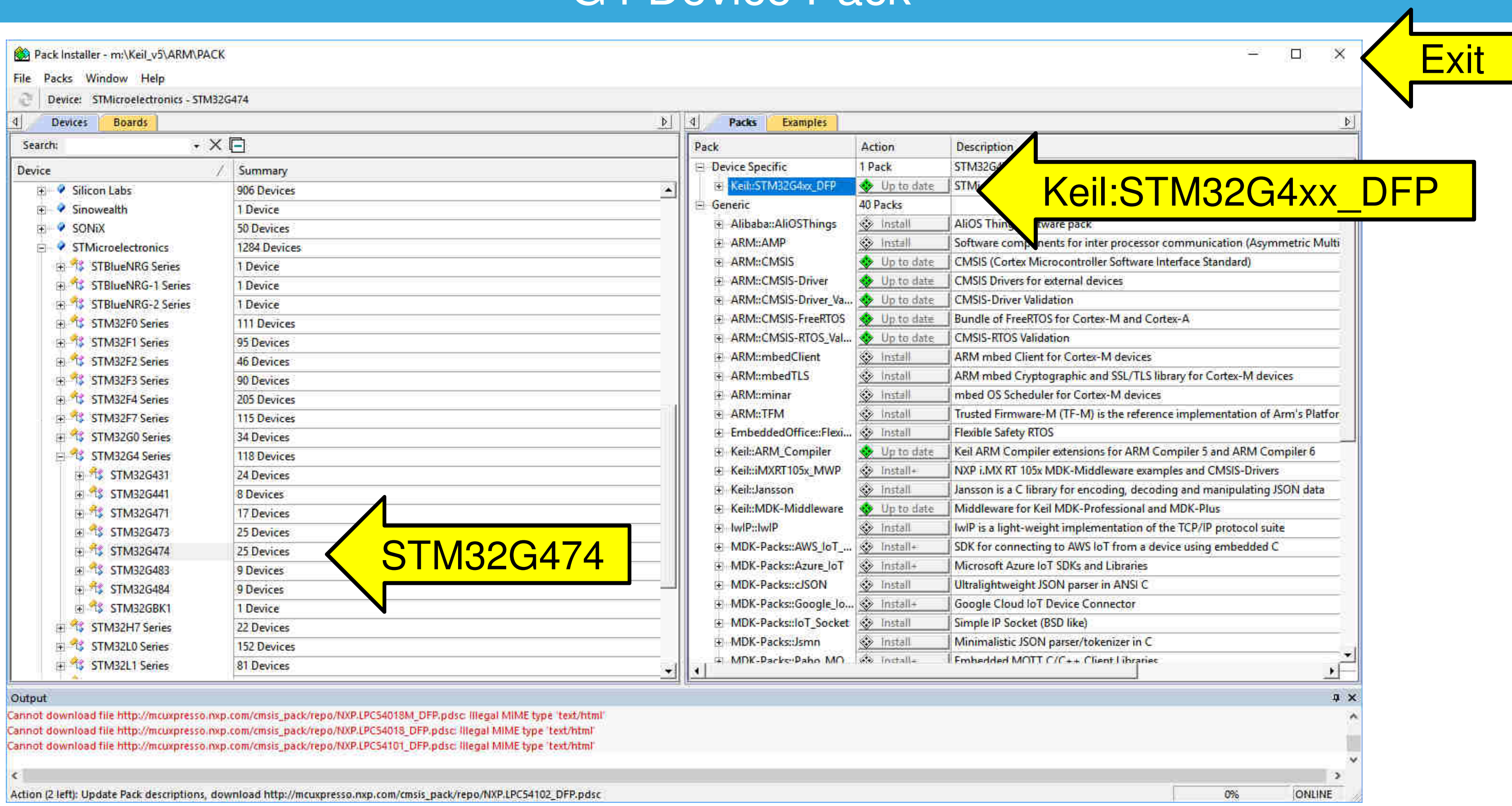
Keil MDK-ARM

Check the Device Pack



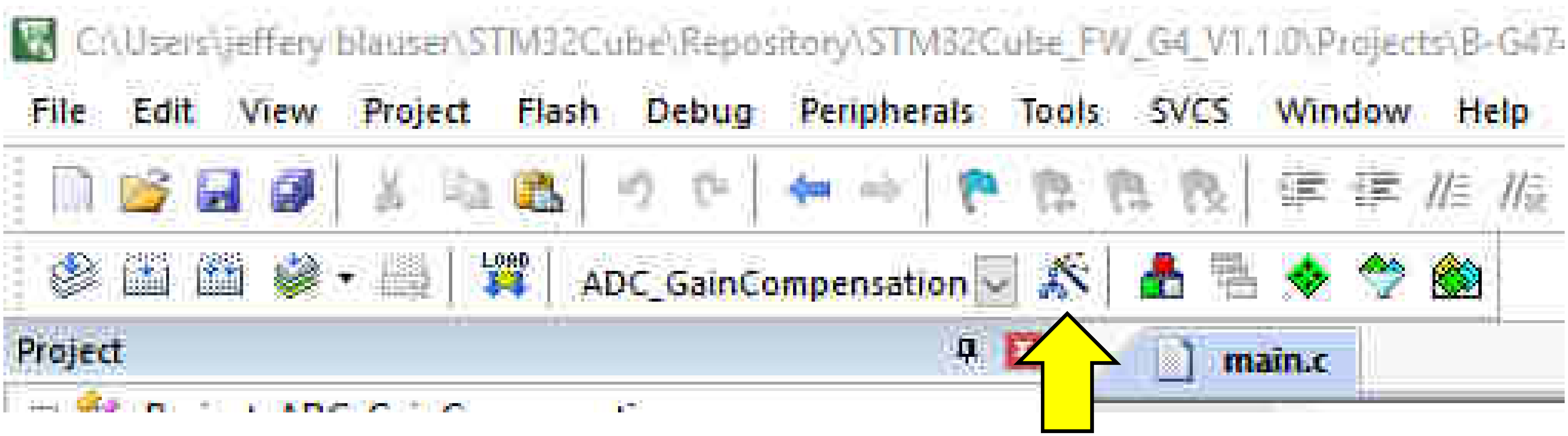
Keil MDK-ARM

G4 Device Pack

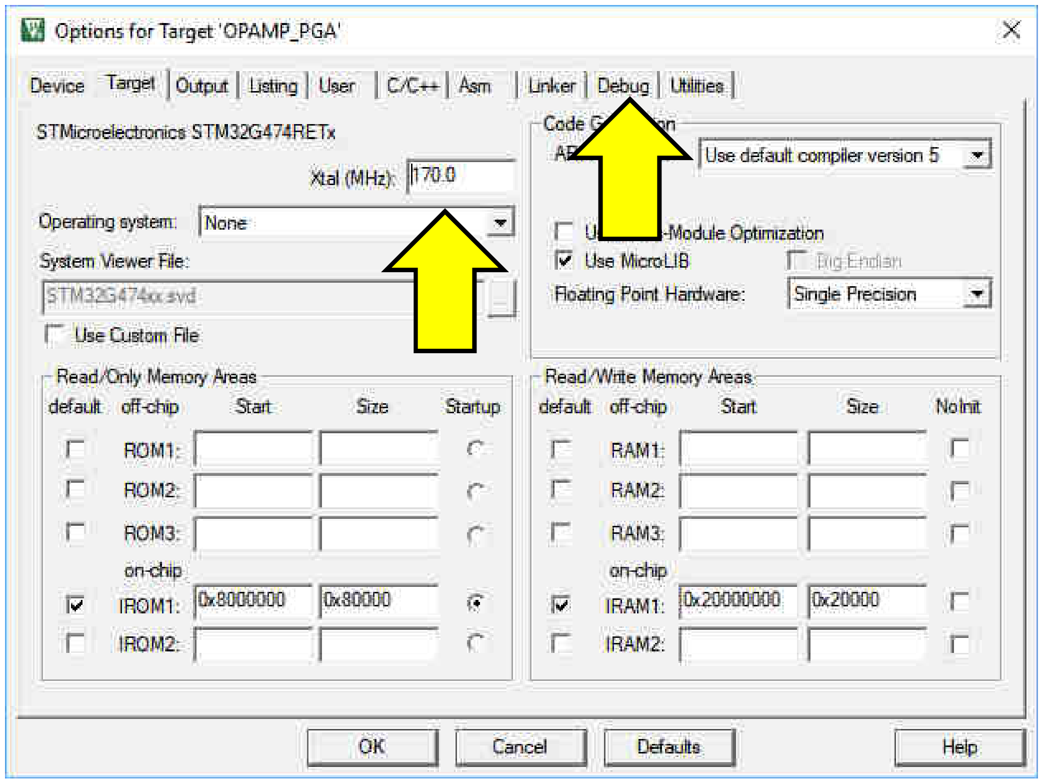


Keil MDK-ARM

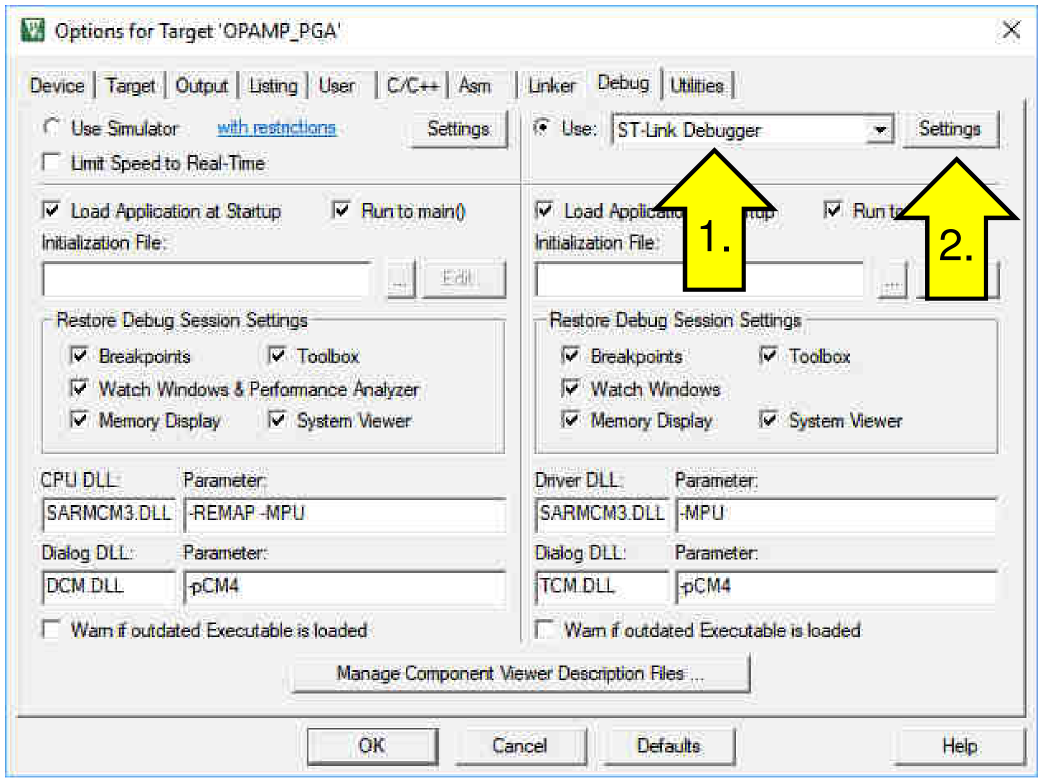
Check the Options



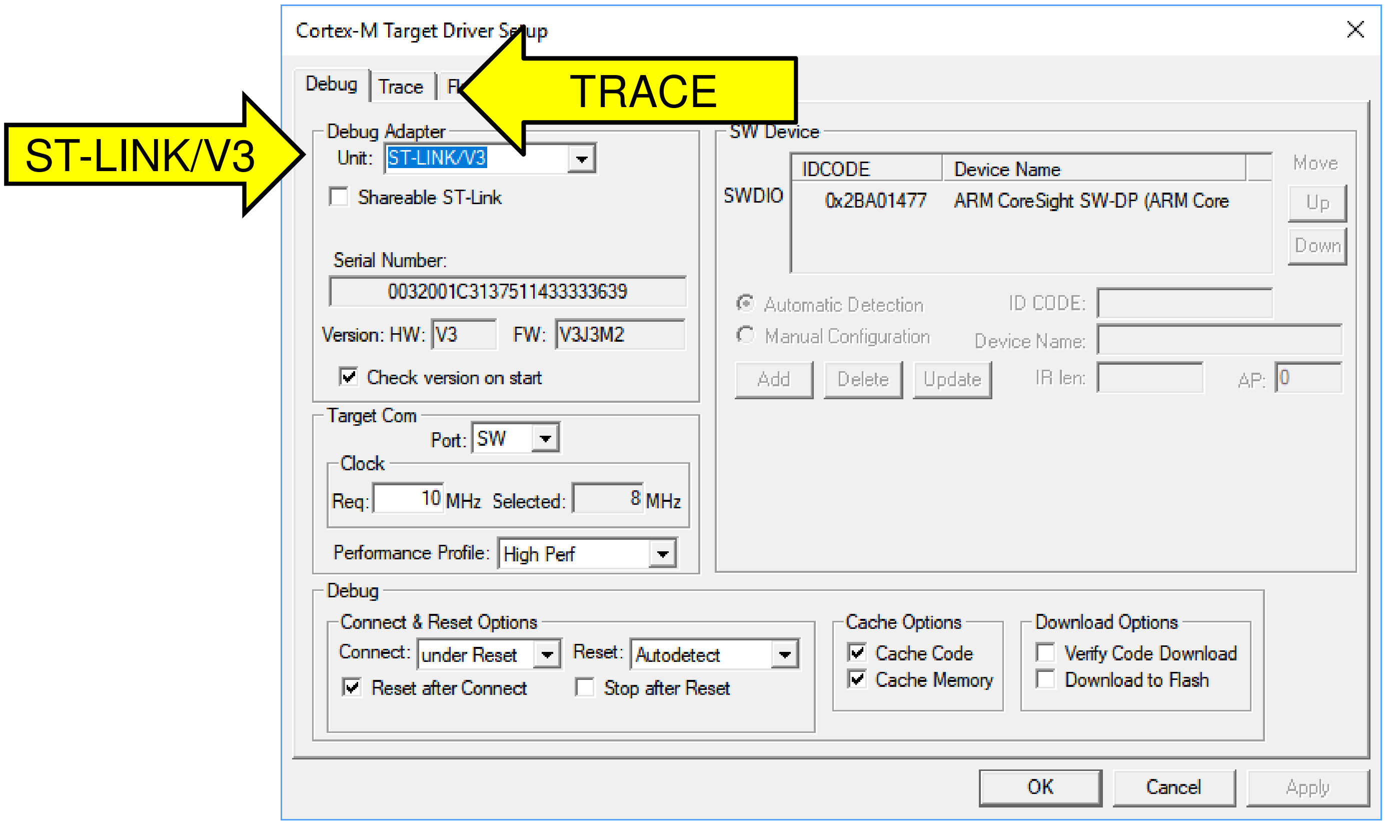
Clock speed and Debug



ST-Link Debugger - Settings

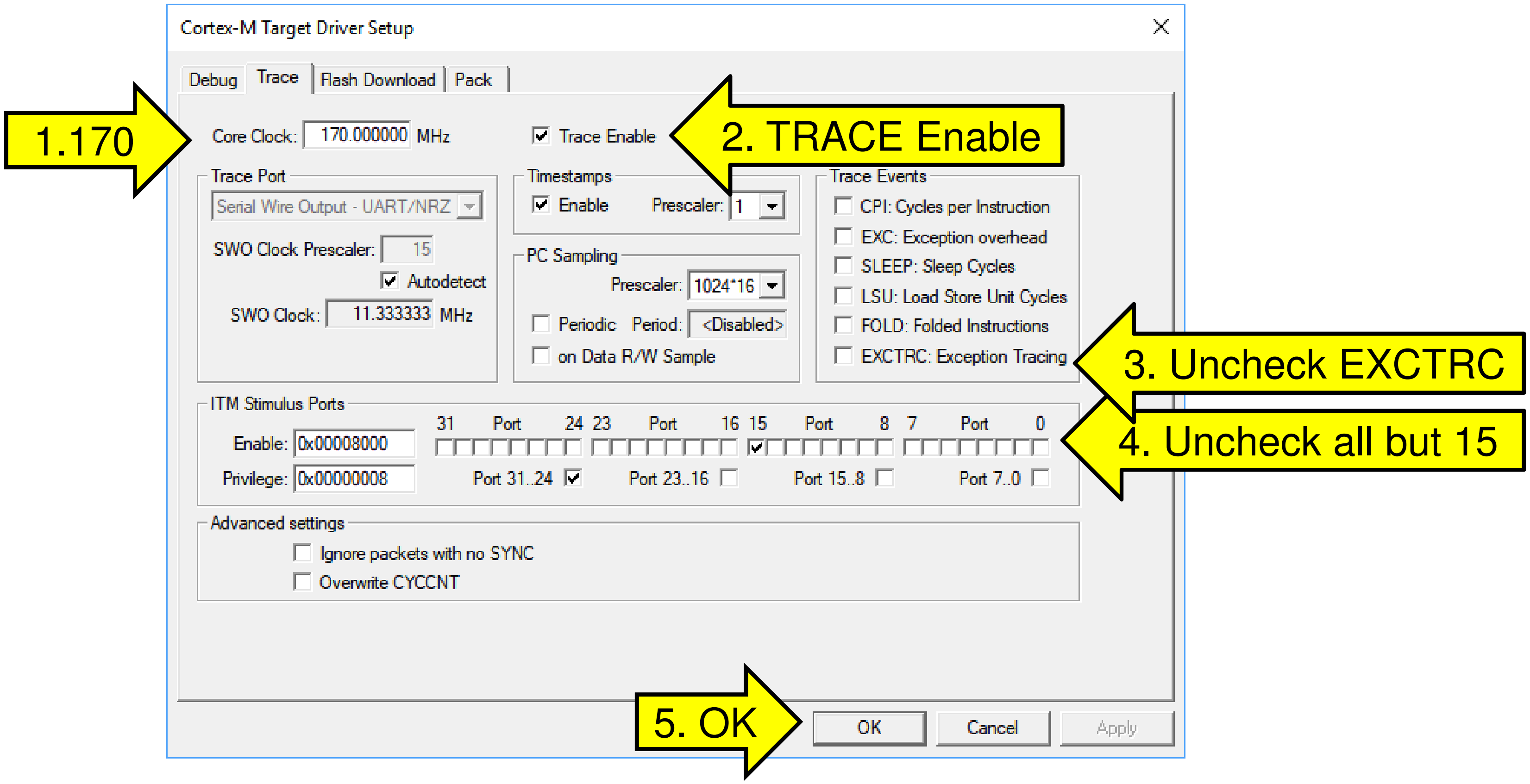


ST-Link/V3 & TRACE



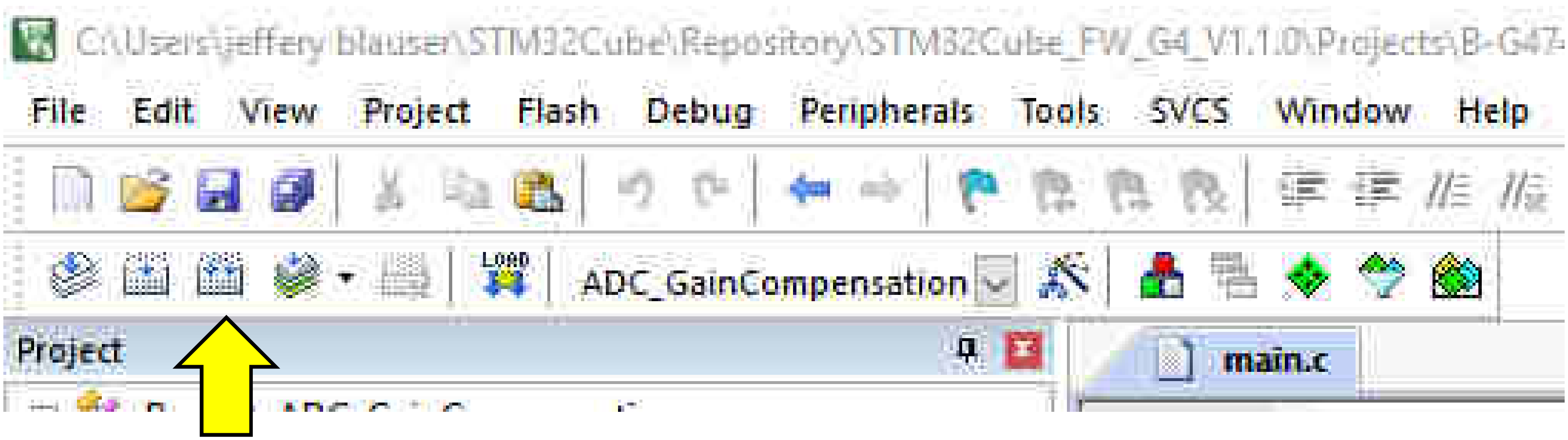
Keil MDK-ARM

Core Clock & TRACE Enable (After)



Keil MDK-ARM

Build

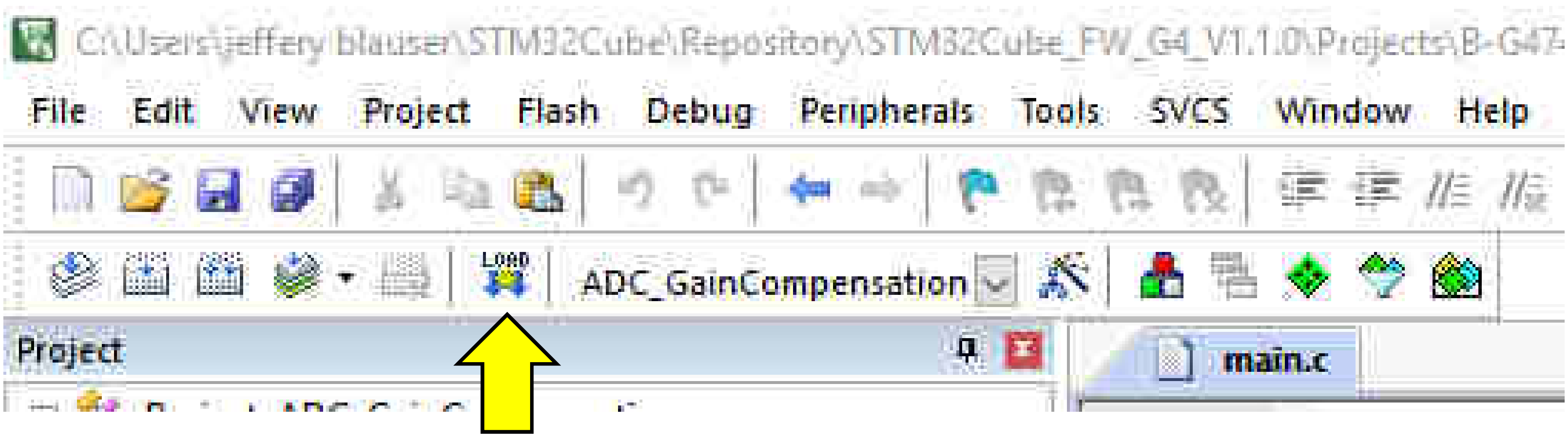


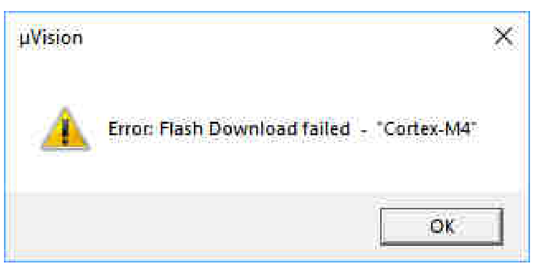
Build Output

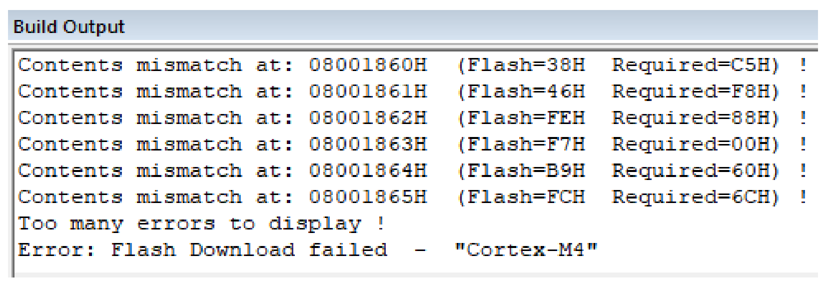
```
Program Size: Code=6516 RO-data=596 RW-data=44 ZI-data=1268
FromELF: creating hex file...
OPAMP_PGA\OPAMP_PGA.hex: Warning: Q9931W: Your license for feature Keil will expire in 30 days
Finished: 0 information, 1 warning and 0 error messages.
"OPAMP_PGA\OPAMP_PGA.axf" - 0 Error(s), 0 Warning(s).
Build Time Elapsed:  00:01:22
```

Keil MDK-ARM

Load

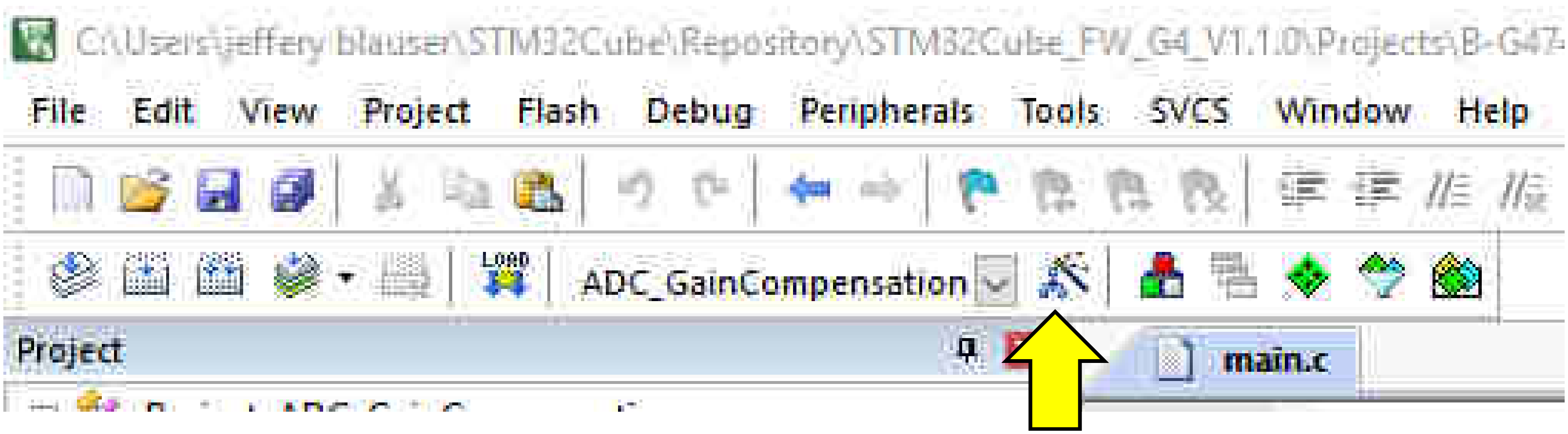




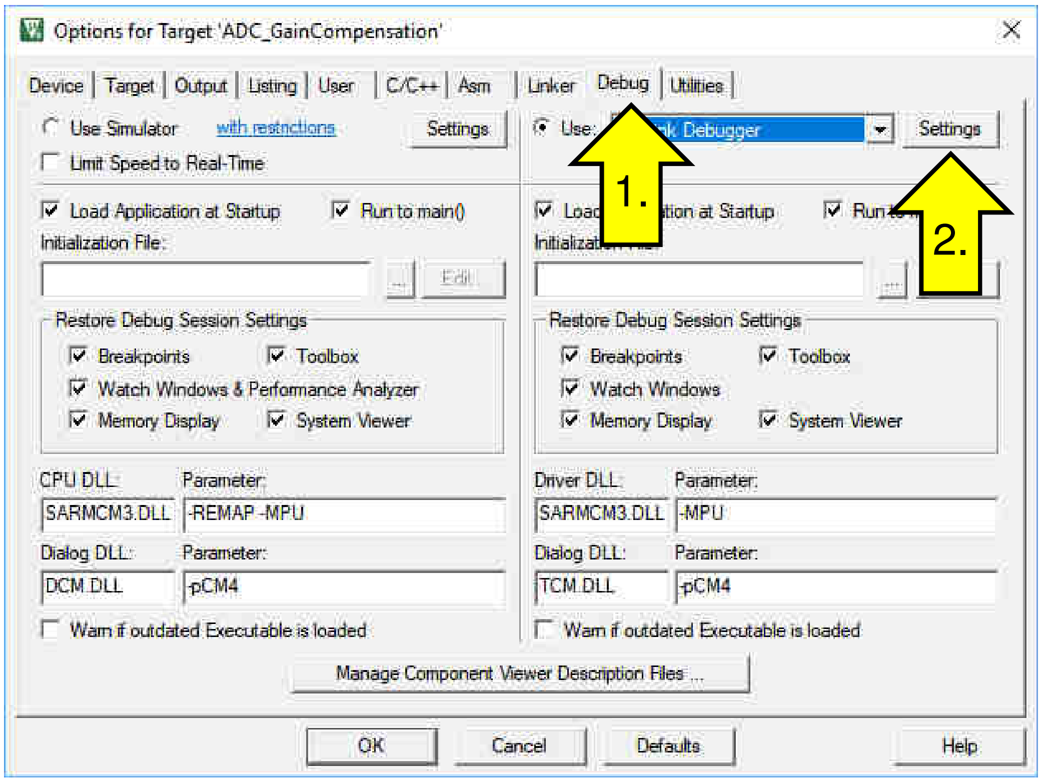


Keil MDK-ARM

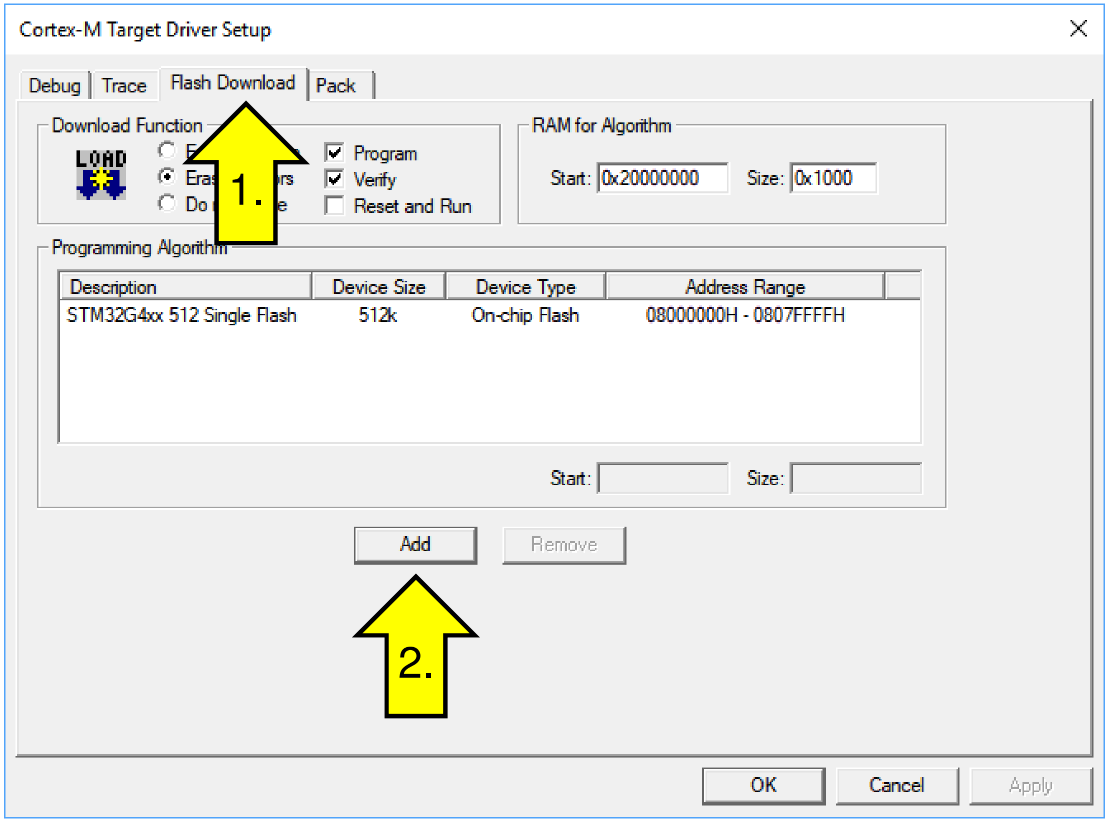
Options



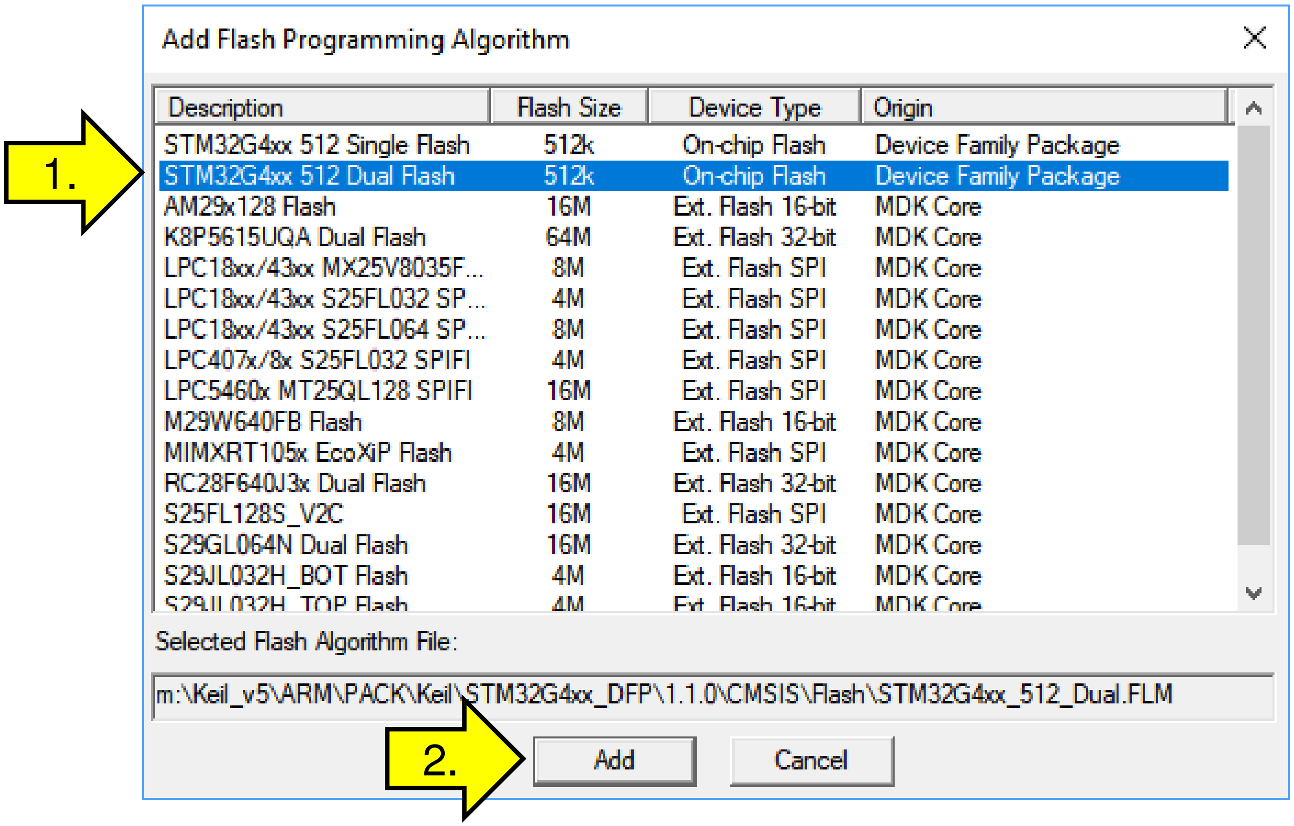
ST-Link Debugger - Settings



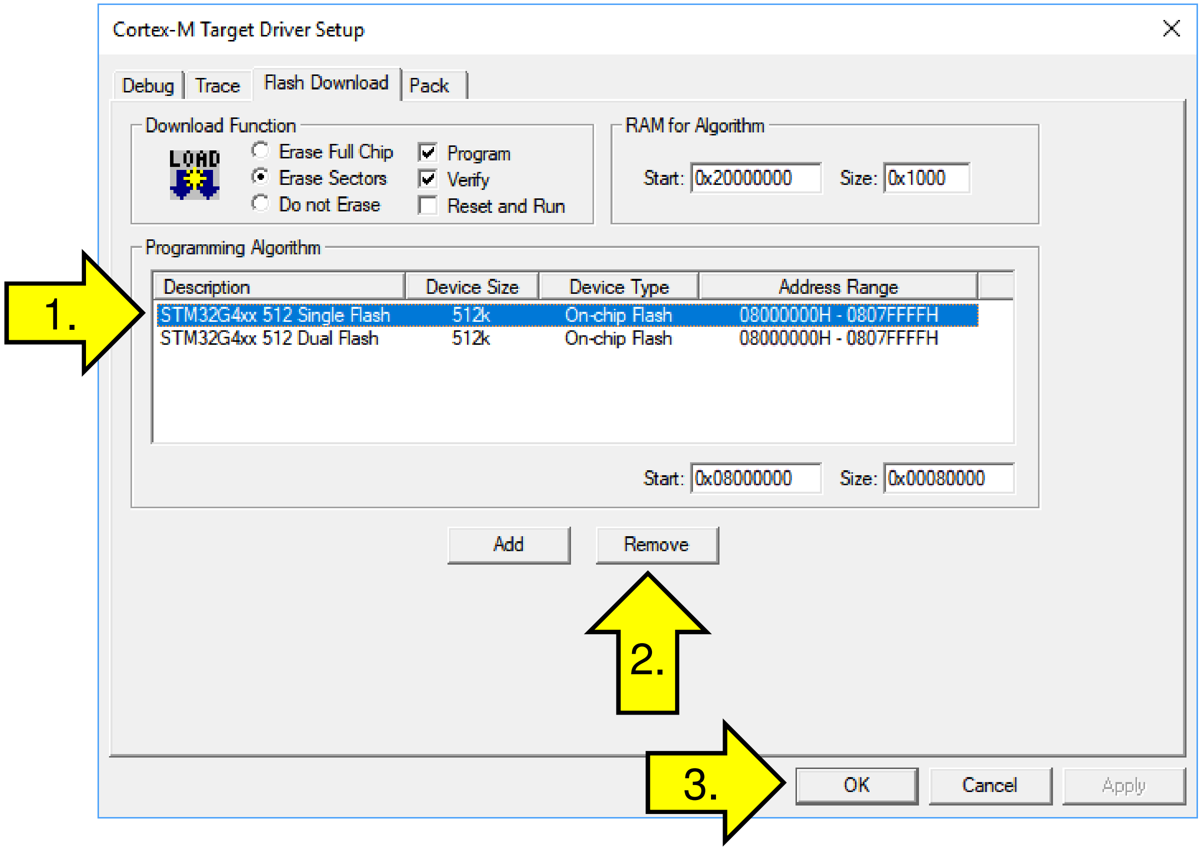
Flash Download → Add



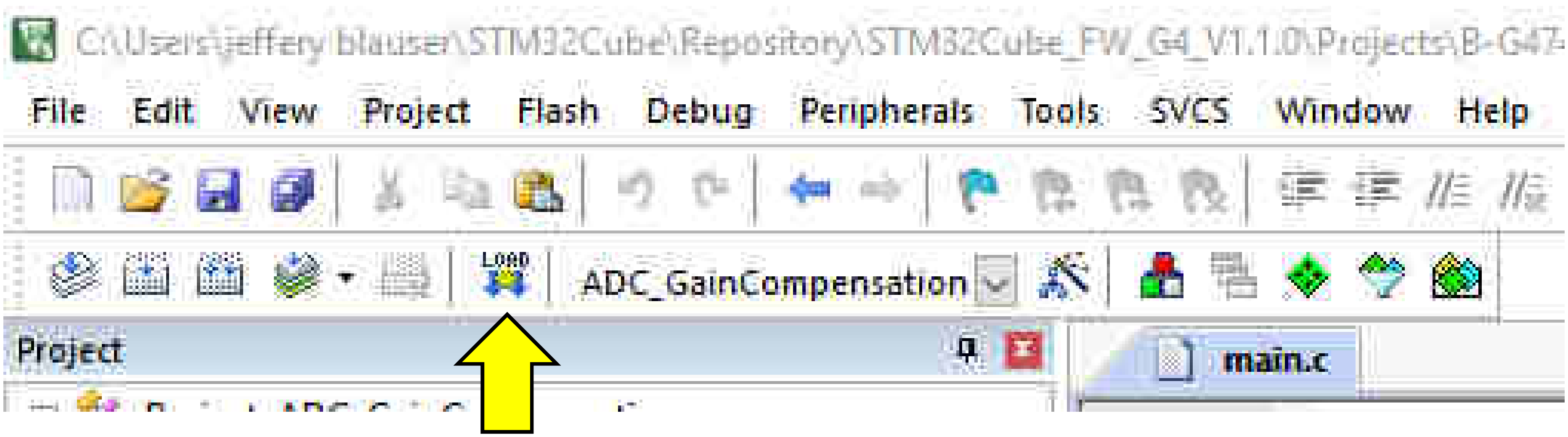
STM32G4xx 512 Dual Flash



Remove Single Flash



Load

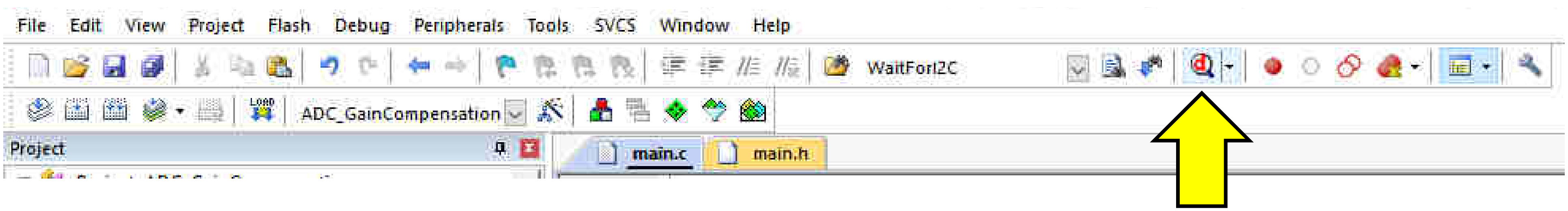


Build Output

```
Build Time Elapsed:  00:01:22
Load "OPAMP_PGA\\OPAMP_PGA.axf"
Erase Done.
Programming Done.
Verify OK.
Flash Load finished at 17:34:57
```

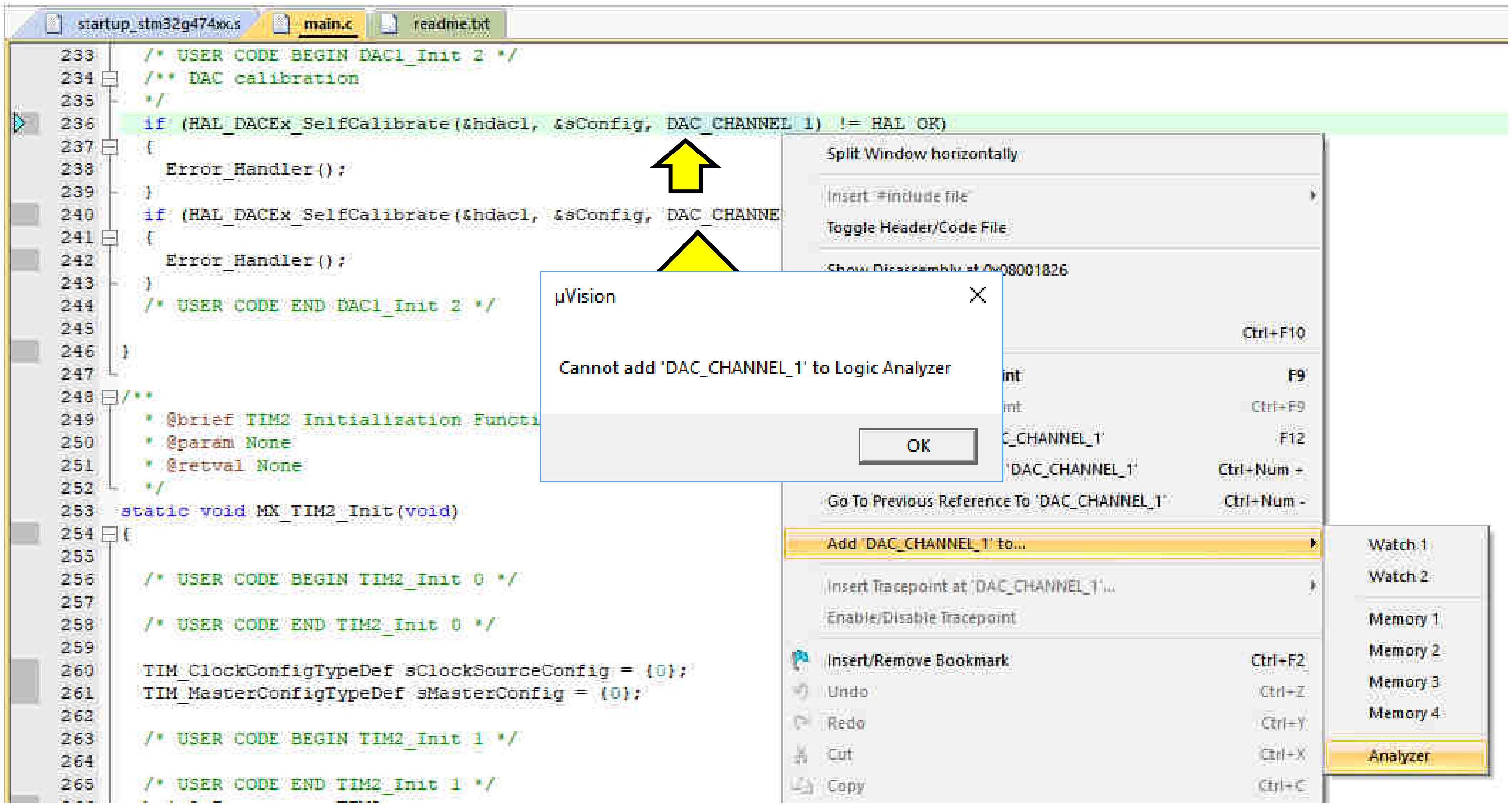
Keil MDK-ARM

Debugger

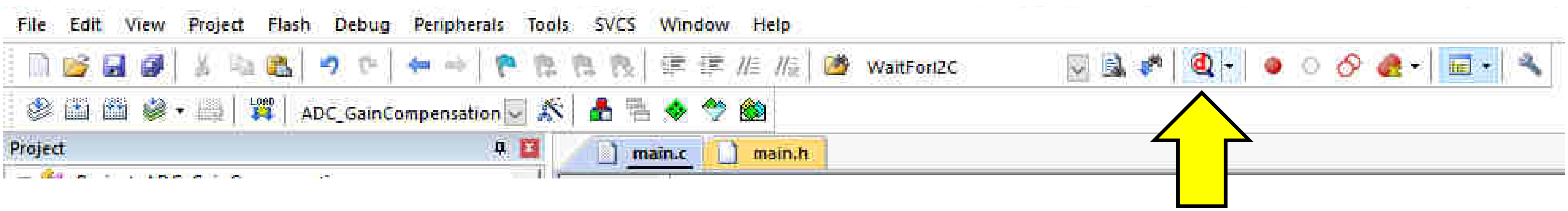


Keil MDK-ARM

Place variable in the Analyzer – main.c about lines 235 & 240



Exit Debugger



Keil MDK-ARM

Add 3 new variables in main.c about line 59

```
54  /* USER CODE BEGIN PV */
55  /* Private variables ---------------------------
56  /* set to 1 after User push-button interrupt  */
57  __IO uint32_t UserButtonStatus = 0;
58
59  uint16_t  Display_DAC1DOR1;
60  uint16_t  Display_DAC3DOR2;
61  uint16_t  Display_ADC3IN1;
```

See CodeLinesForOpAmp_09032019.txt in
c:\2019_STM32G4_Workshop\HandsOn_Labs\OPAMP

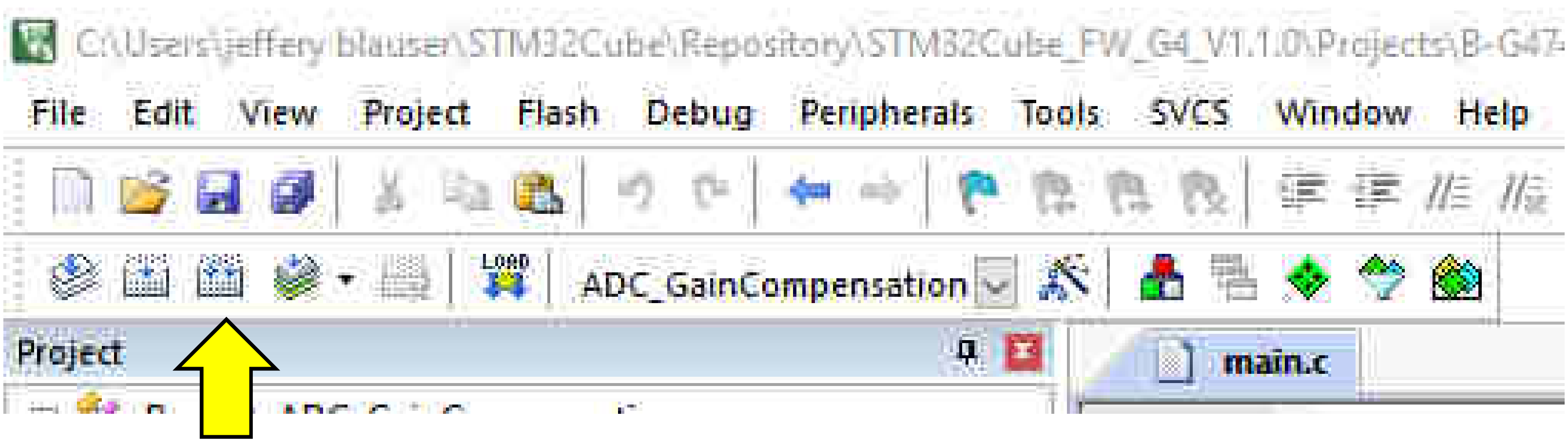
Keil MDK-ARM

Add new lines in main.c about line 177

```
174         Error_Handler();
175       }
176     }
177     /* Display the global variables in the Logic Analyzer */
178     Display_DAC1DOR1 = DAC1->DOR1;
179     HAL_Delay(1);   //1 millisecond delay to transfer data to the PC
180     Display_DAC3DOR2 = DAC3->DOR1;
181     HAL_Delay(1);   //1 millisecond delay to transfer data to the PC
182     Display_ADC3IN1 = ADC3->DR;
183     HAL_Delay(1);   //1 millisecond delay to transfer data to the PC
184   }
185     /* USER CODE END WHILE */
```

See CodeLinesForOpAmp_09032019.txt in
c:\2019_STM32G4_Workshop\HandsOn_Labs\OPAMP

Build

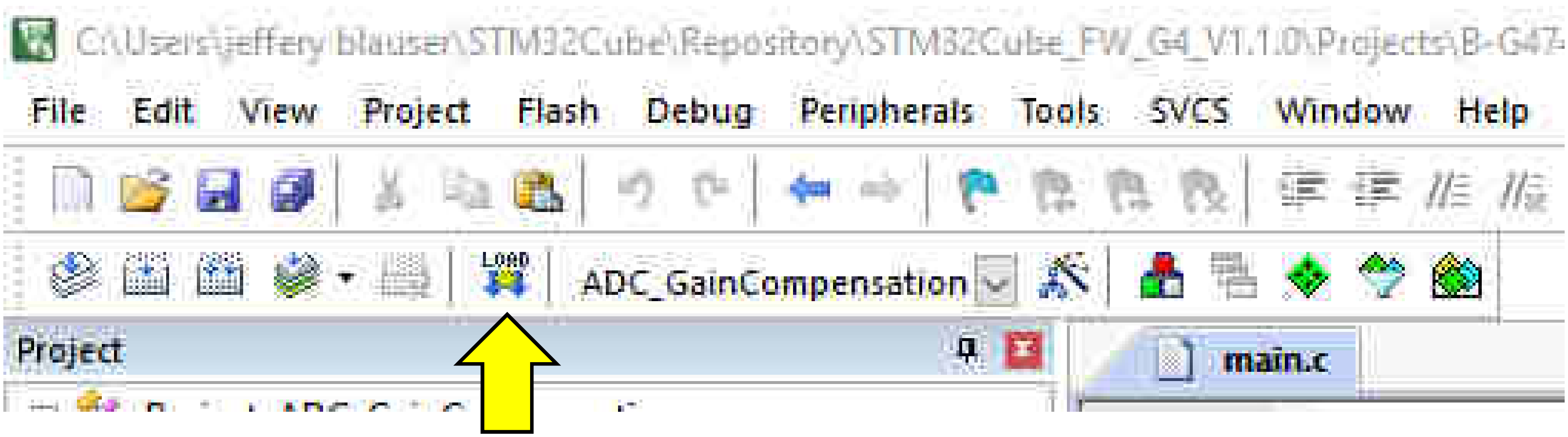


Build Output

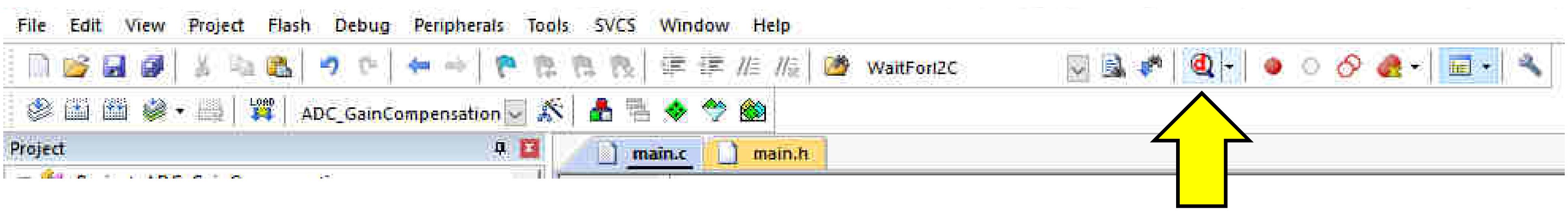
```
Program Size: Code=6516 RO-data=596 RW-data=44 ZI-data=1268
FromELF: creating hex file...
OPAMP_PGA\OPAMP_PGA.hex: Warning: Q9931W: Your license for feature Keil will expire in 30 days
Finished: 0 information, 1 warning and 0 error messages.
"OPAMP_PGA\OPAMP_PGA.axf" - 0 Error(s), 0 Warning(s).
Build Time Elapsed:  00:01:22
```

Keil MDK-ARM

Load



Debugger – Exit back to the Editor

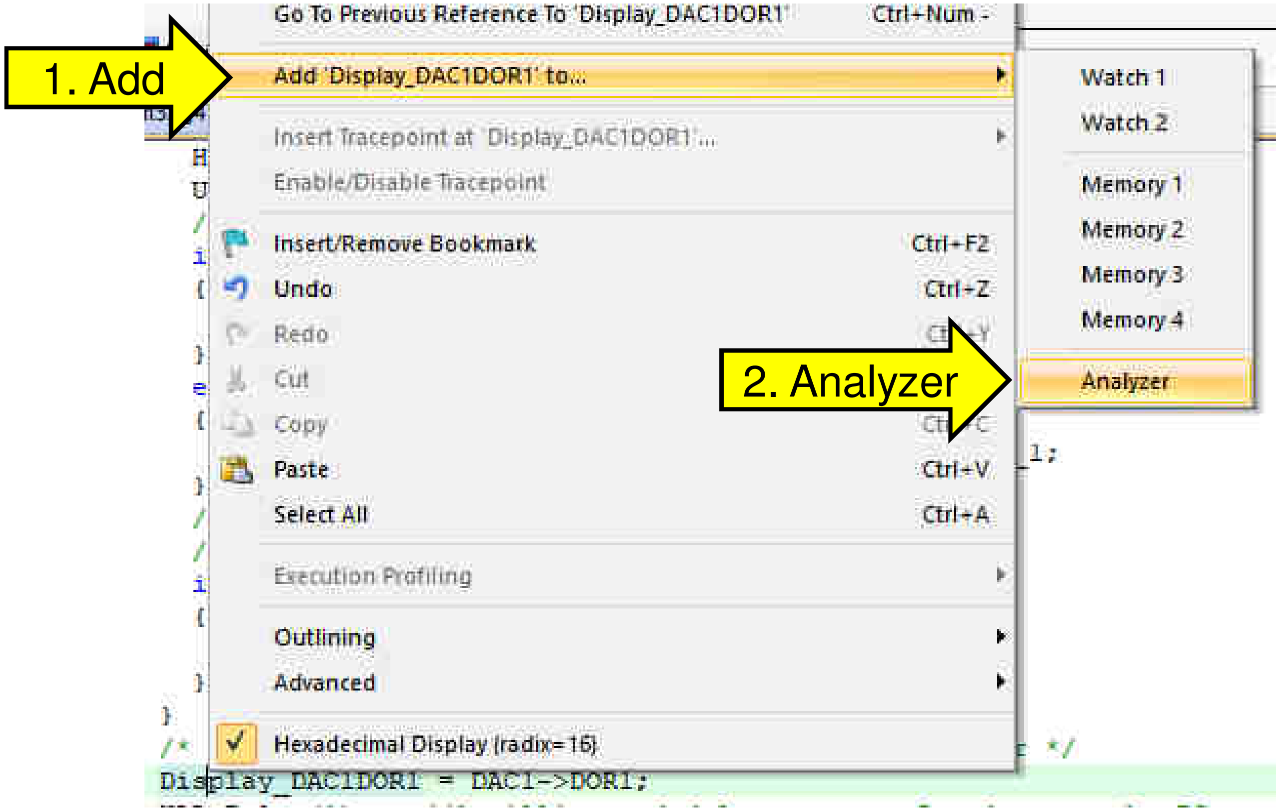


Display_DAC1DOR1 in the Logic Analyzer

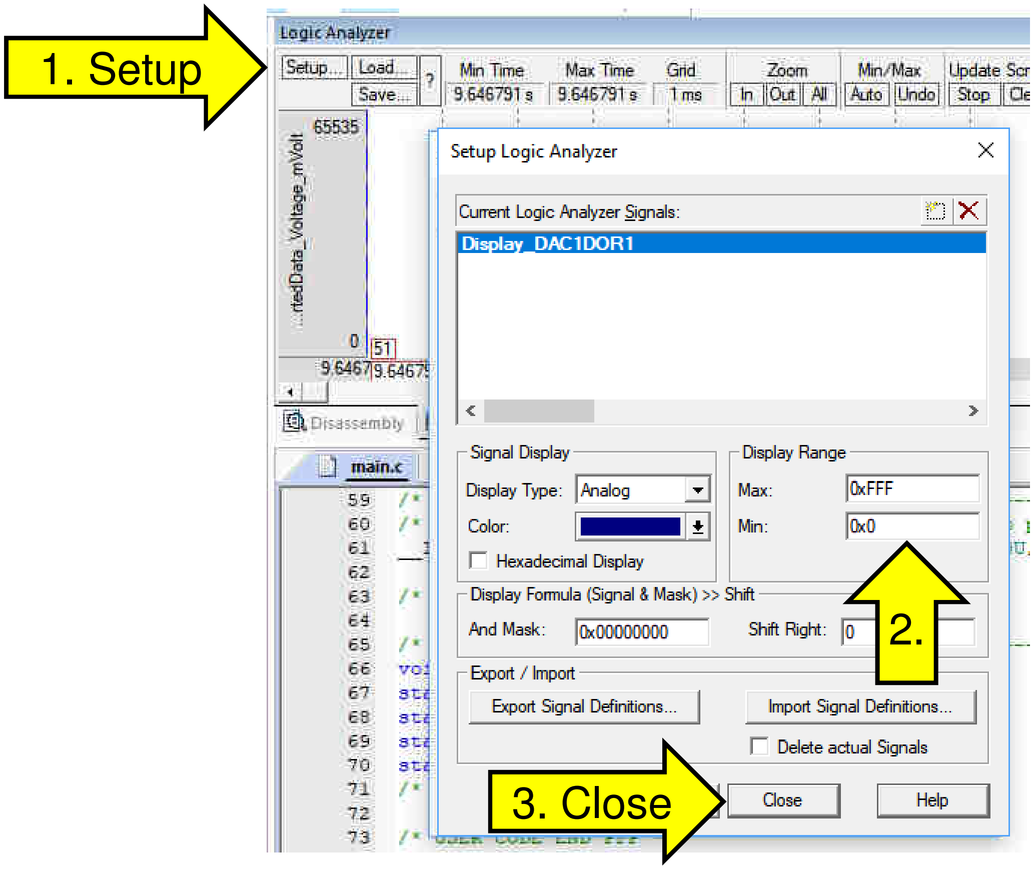
Right Click

```
            Error_Handler();
          }
        }
        /* Display the global variables in the Logic Analyzer */
        Display_DAC1DOR1 = DAC1->DOR1;
        HAL_Delay(1);    //1 millisecond delay to transfer data to the PC
        Display_DAC3DOR2 = DAC3->DOR1;
        HAL_Delay(1);    //1 millisecond delay to transfer data to the PC
        Display_ADC3IN1 = ADC3->DR;
        HAL_Delay(1);    //1 millisecond delay to transfer data to the PC
      }
        /* USER CODE END WHILE */
```

Display_DAC1DOR1 in the Logic Analyzer

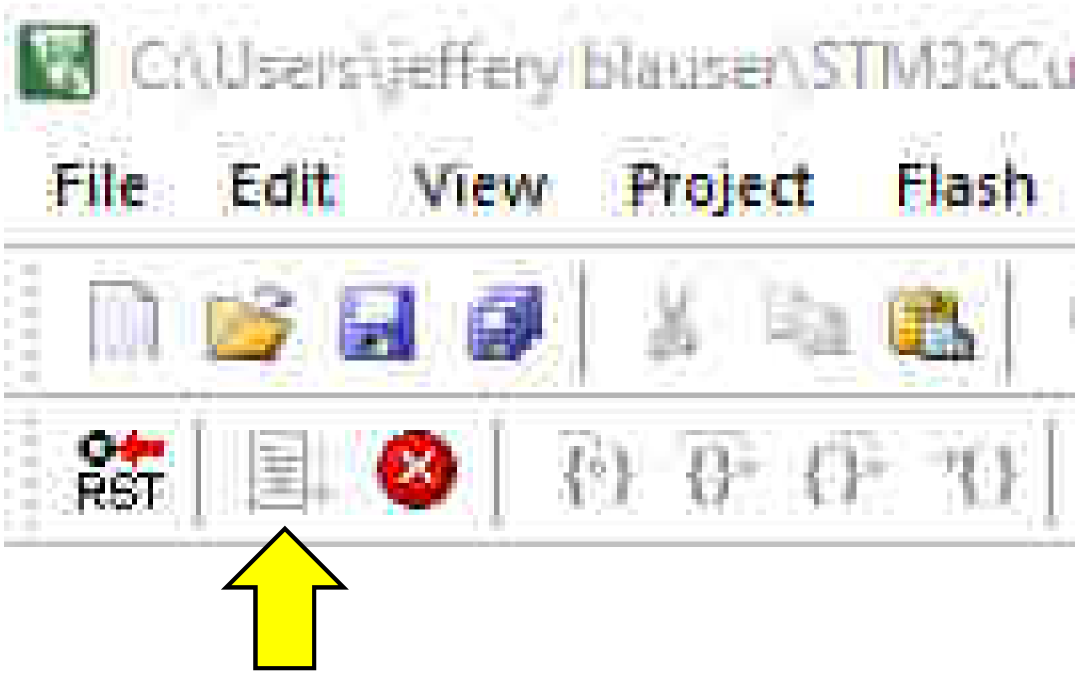


Setup – Min=0x0, Max = 0xFFF

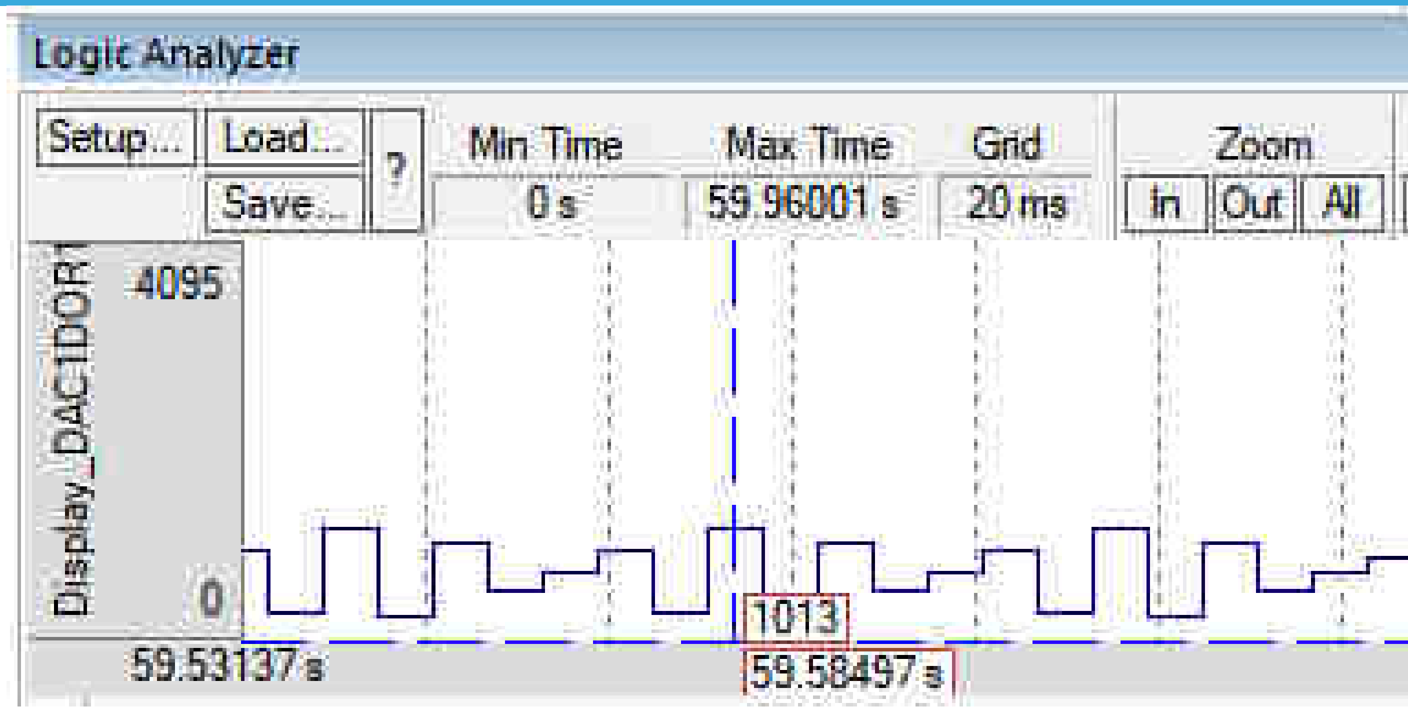


Keil MDK-ARM

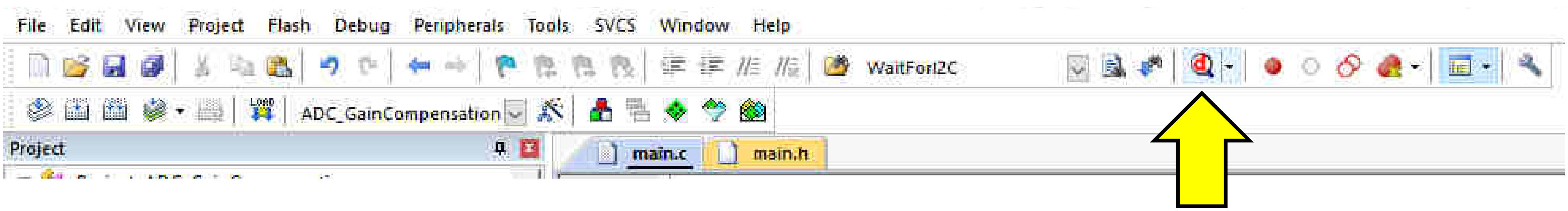
RUN



DAC1_OUTPUT1

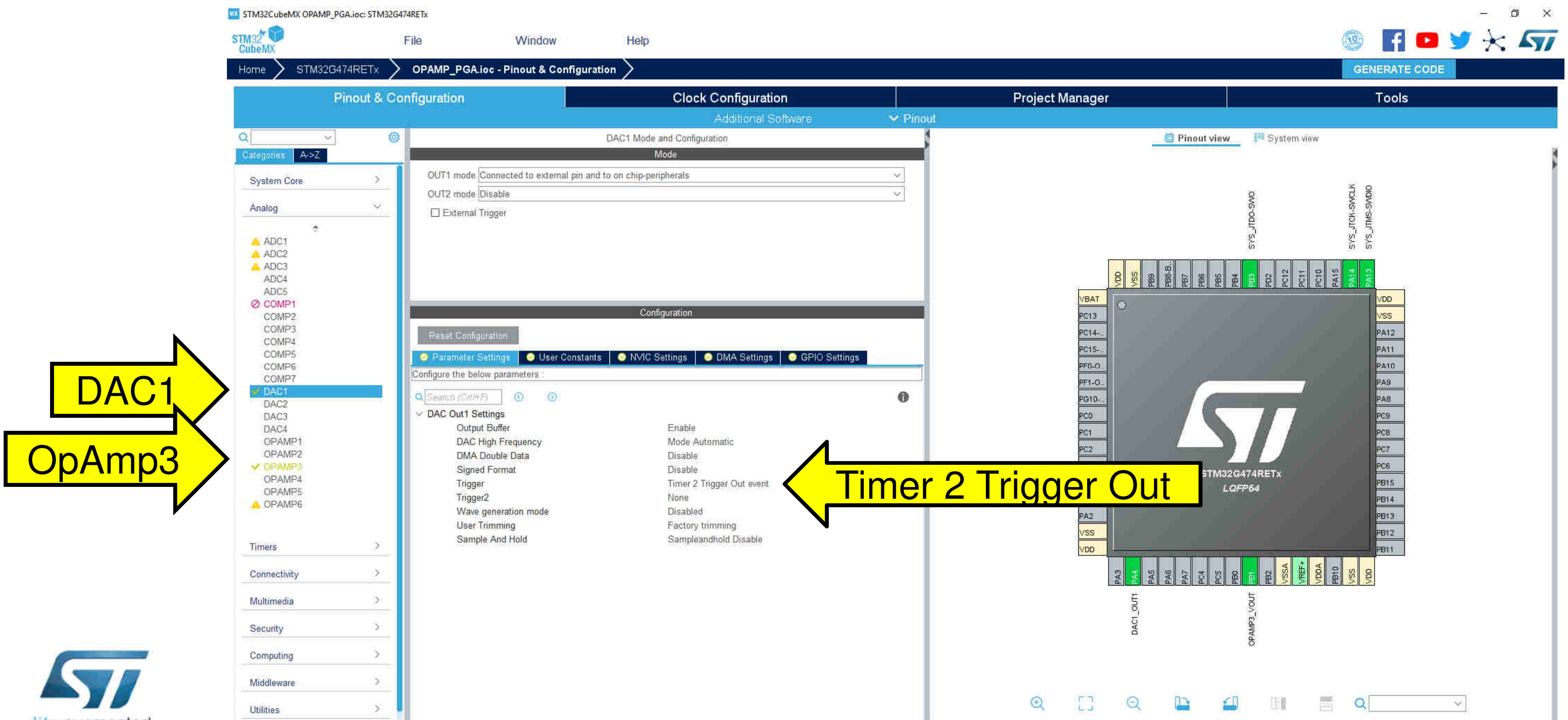


Debugger – Exit back to the Editor



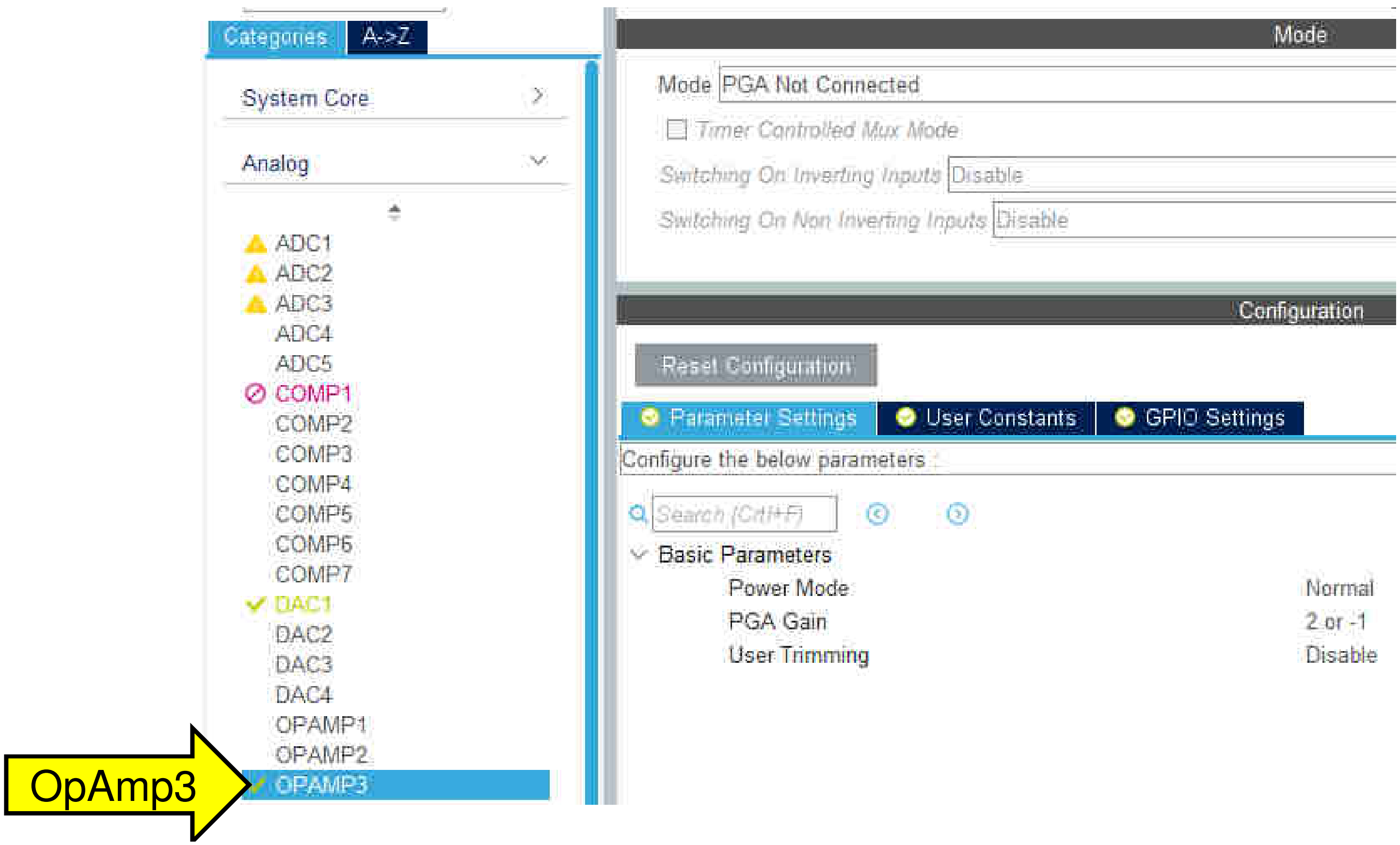
STM32CubeMx

Recall we are using, Op Amp 3



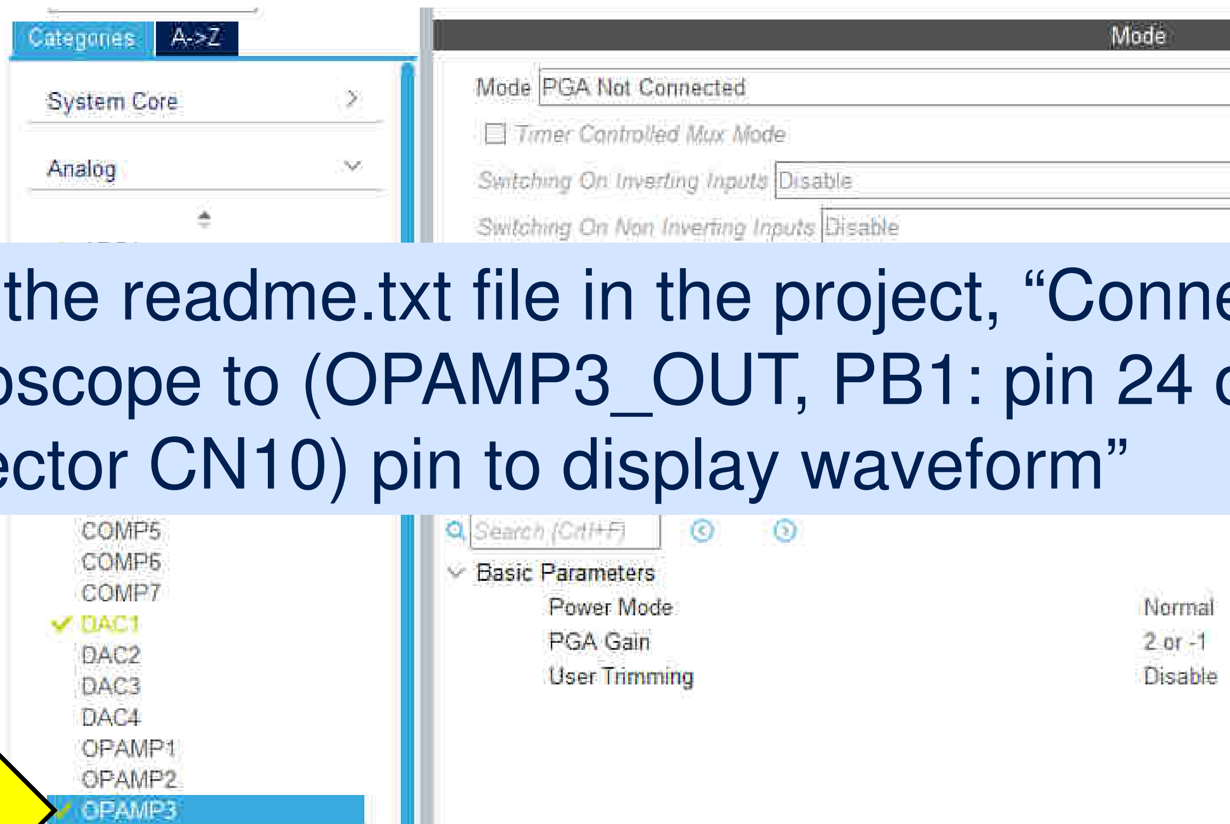
STM32CubeMx

We will go with this for now, but where is the output?



STM32CubeMx

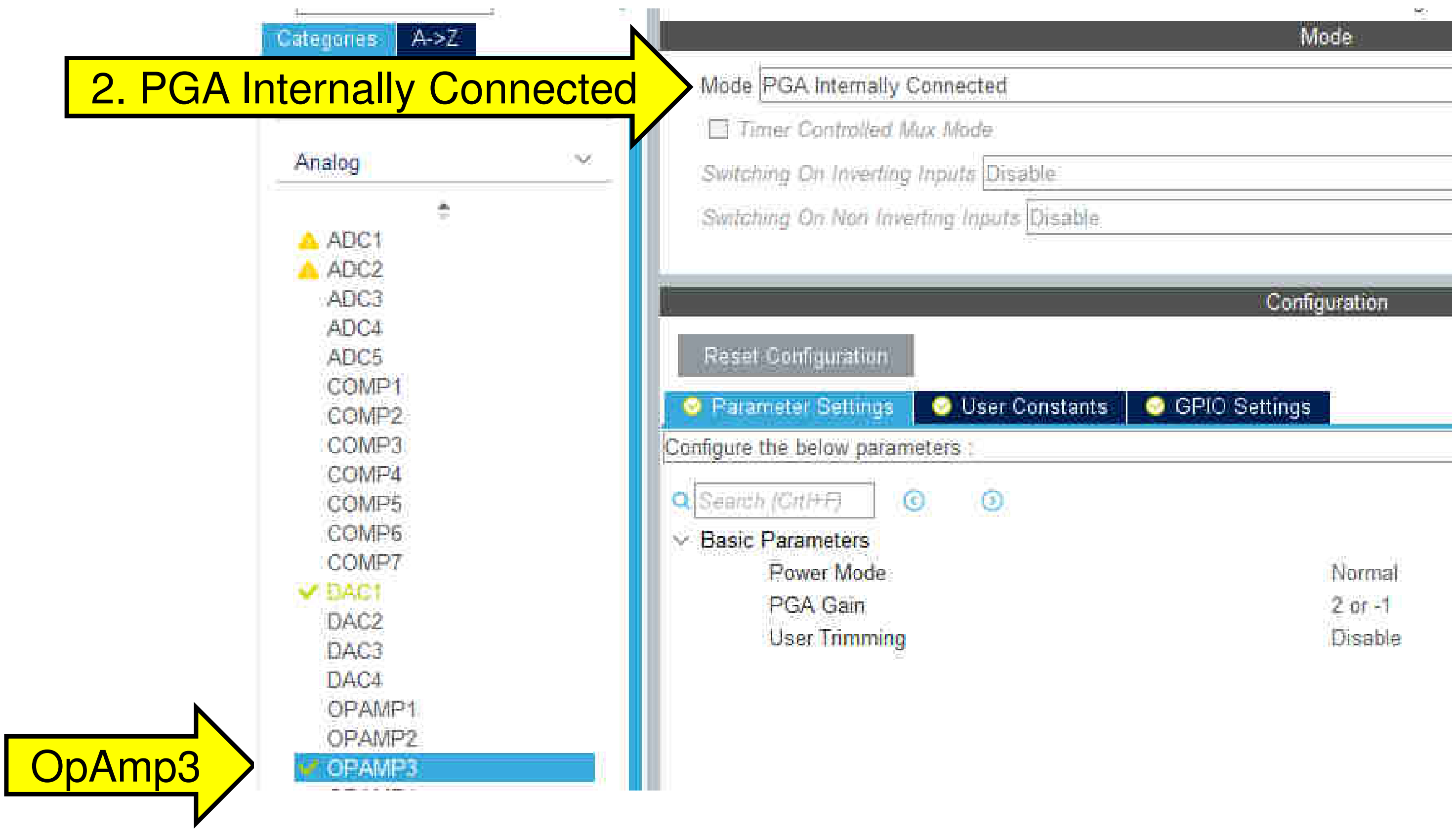
We will go with this for now, but where is the output?



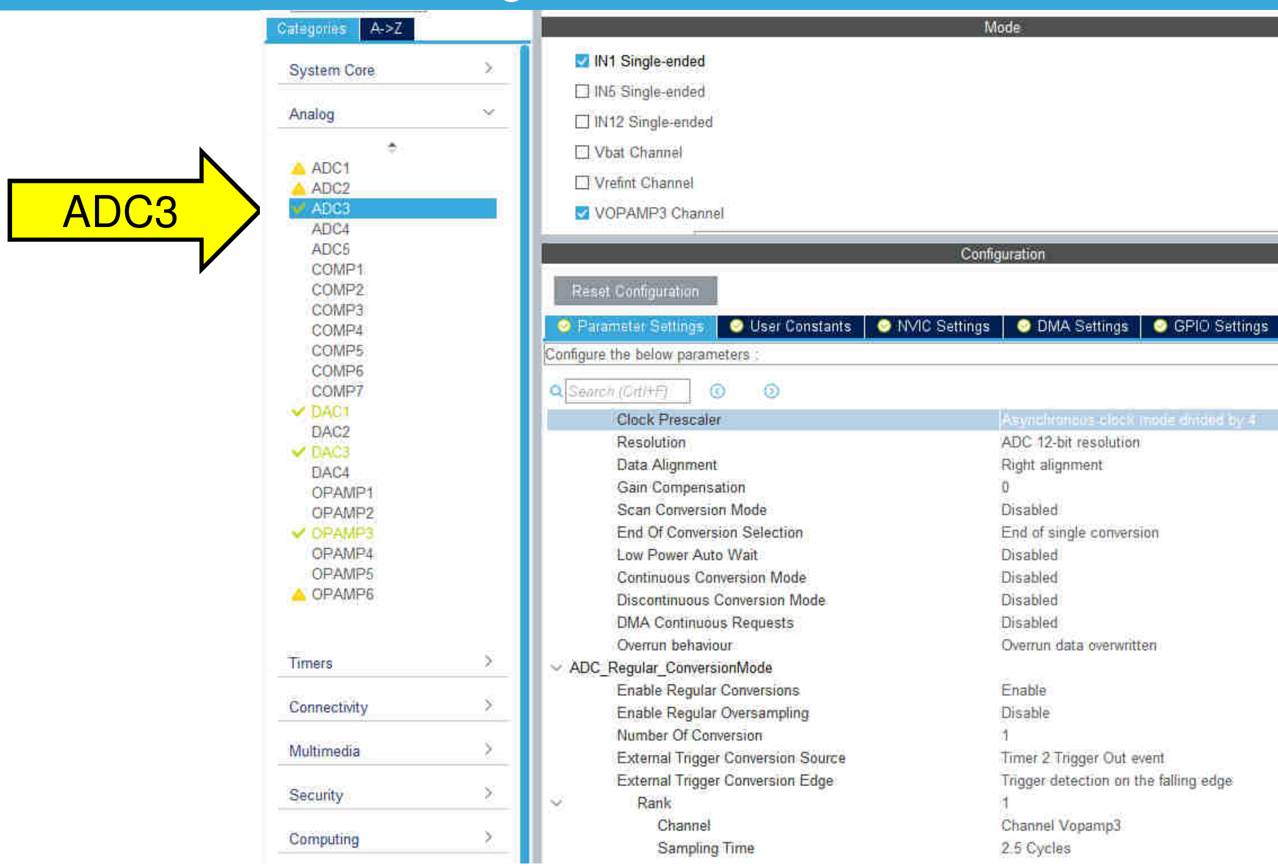
From the readme.txt file in the project, “Connect an oscilloscope to (OPAMP3_OUT, PB1: pin 24 on connector CN10) pin to display waveform”

OpAmp3

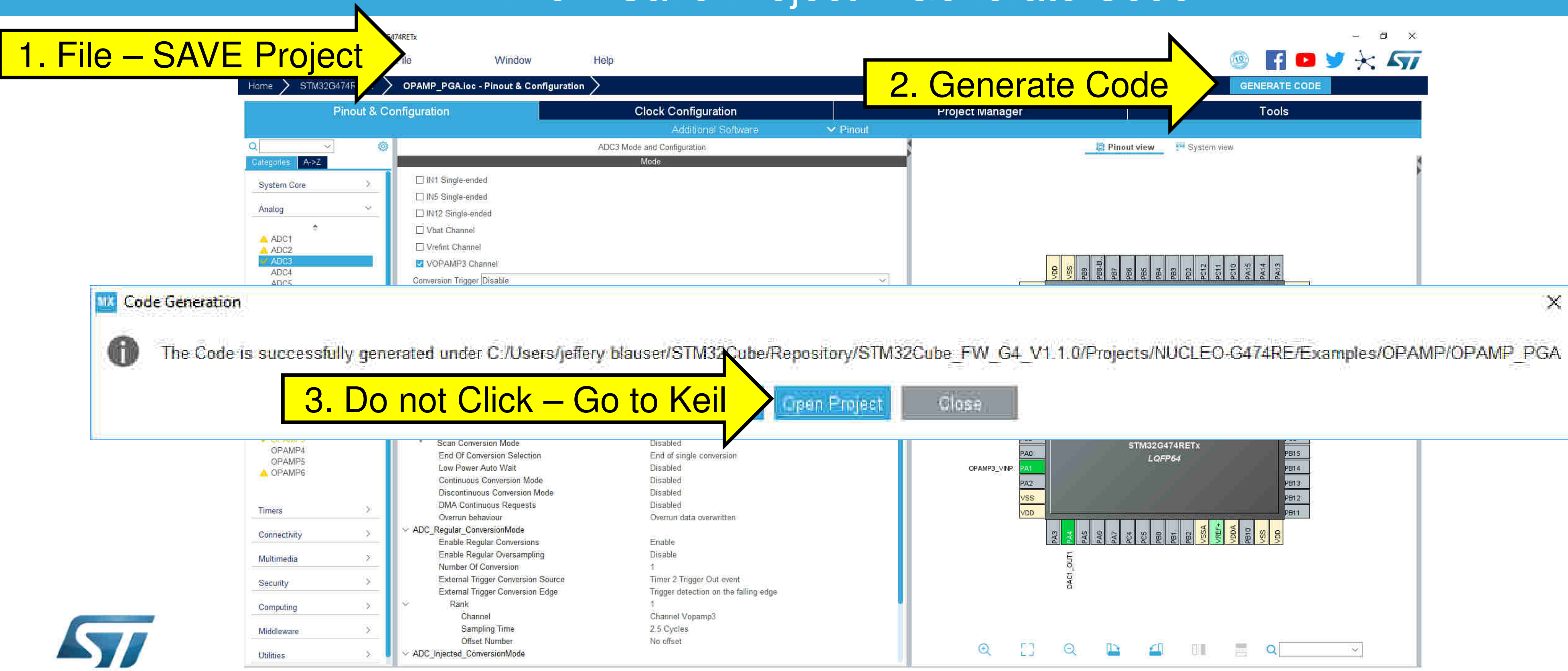
Set the PGA to an internal connection



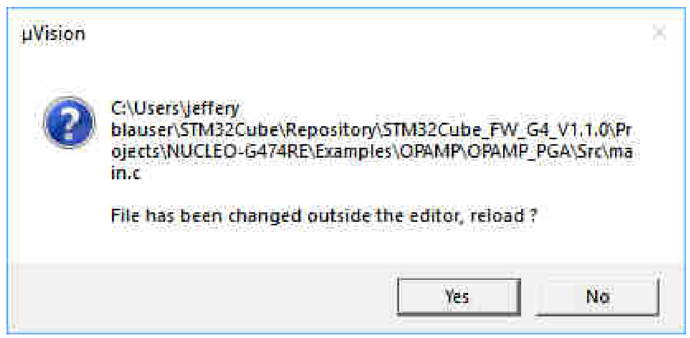
Configure ADC3 – VOPAMP3, etc.



File – Save Project – Generate Code

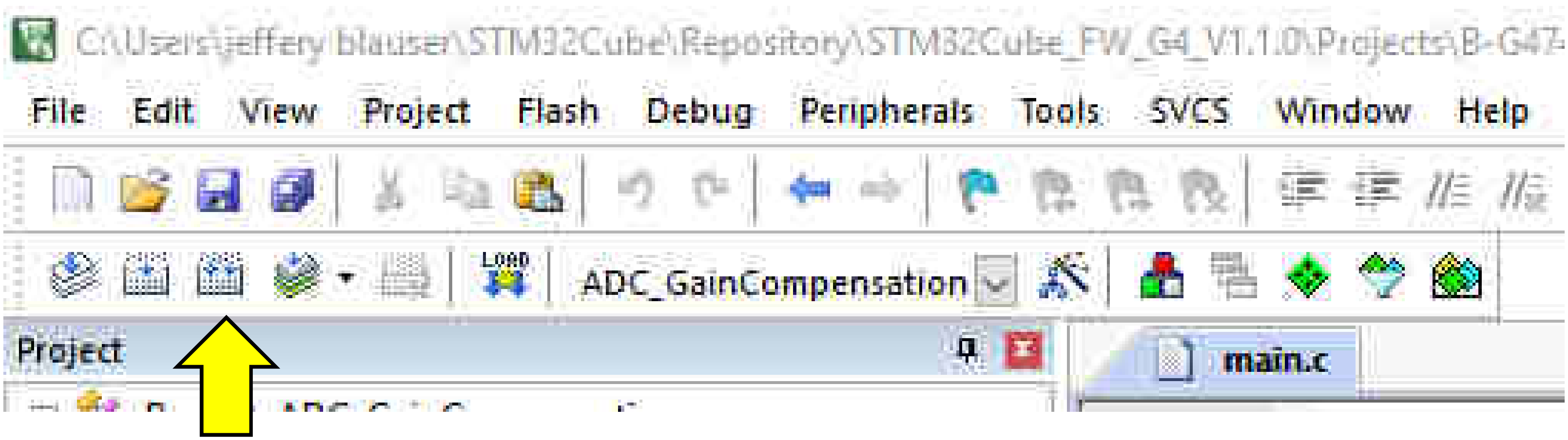


Click YES 3x


µVision
C:\Users\jeffery
blauser\STM32Cube\Repository\STM32Cube_FW_G4_V1.1.0\Pr
ojects\NUCLEO-G474RE\Examples\OPAMP\OPAMP_PGA\Src\ma
in.c
File has been changed outside the editor, reload ?
Yes
No

Keil MDK-ARM

Build

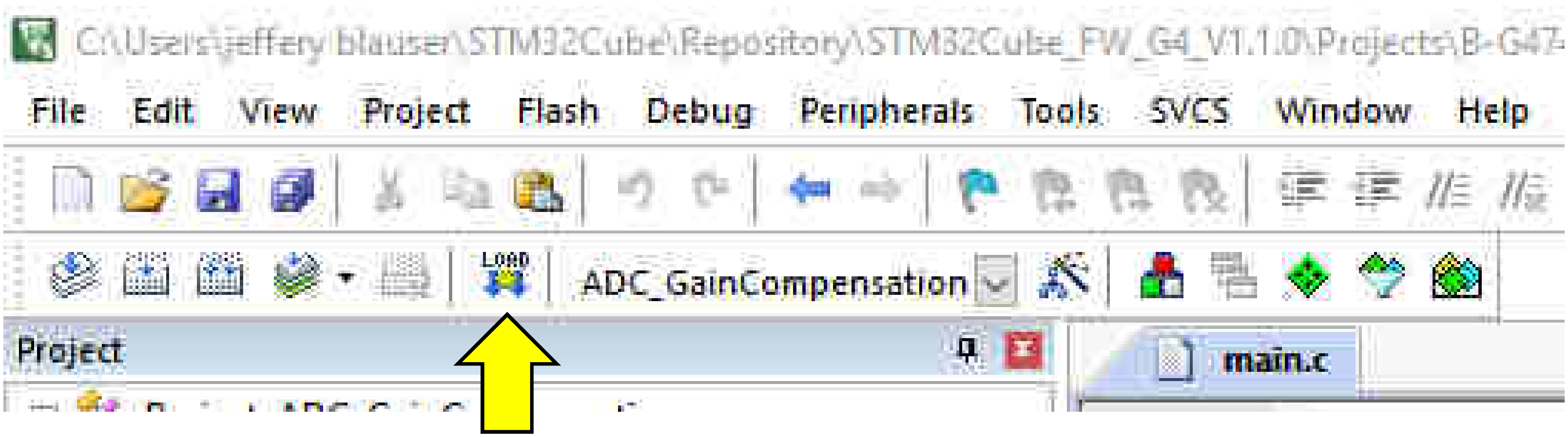


Build Output

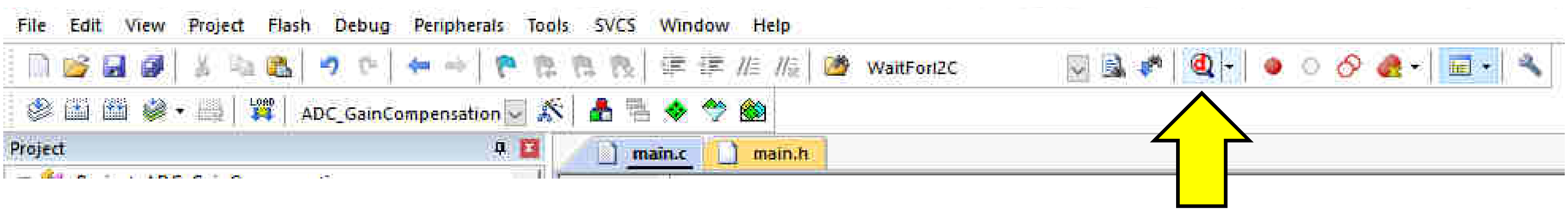
```
Program Size: Code=6516 RO-data=596 RW-data=44 ZI-data=1268
FromELF: creating hex file...
OPAMP_PGA\OPAMP_PGA.hex: Warning: Q9931W: Your license for feature Keil will expire in 30 days
Finished: 0 information, 1 warning and 0 error messages.
"OPAMP_PGA\OPAMP_PGA.axf" - 0 Error(s), 0 Warning(s).
Build Time Elapsed:  00:01:22
```

life.augmented

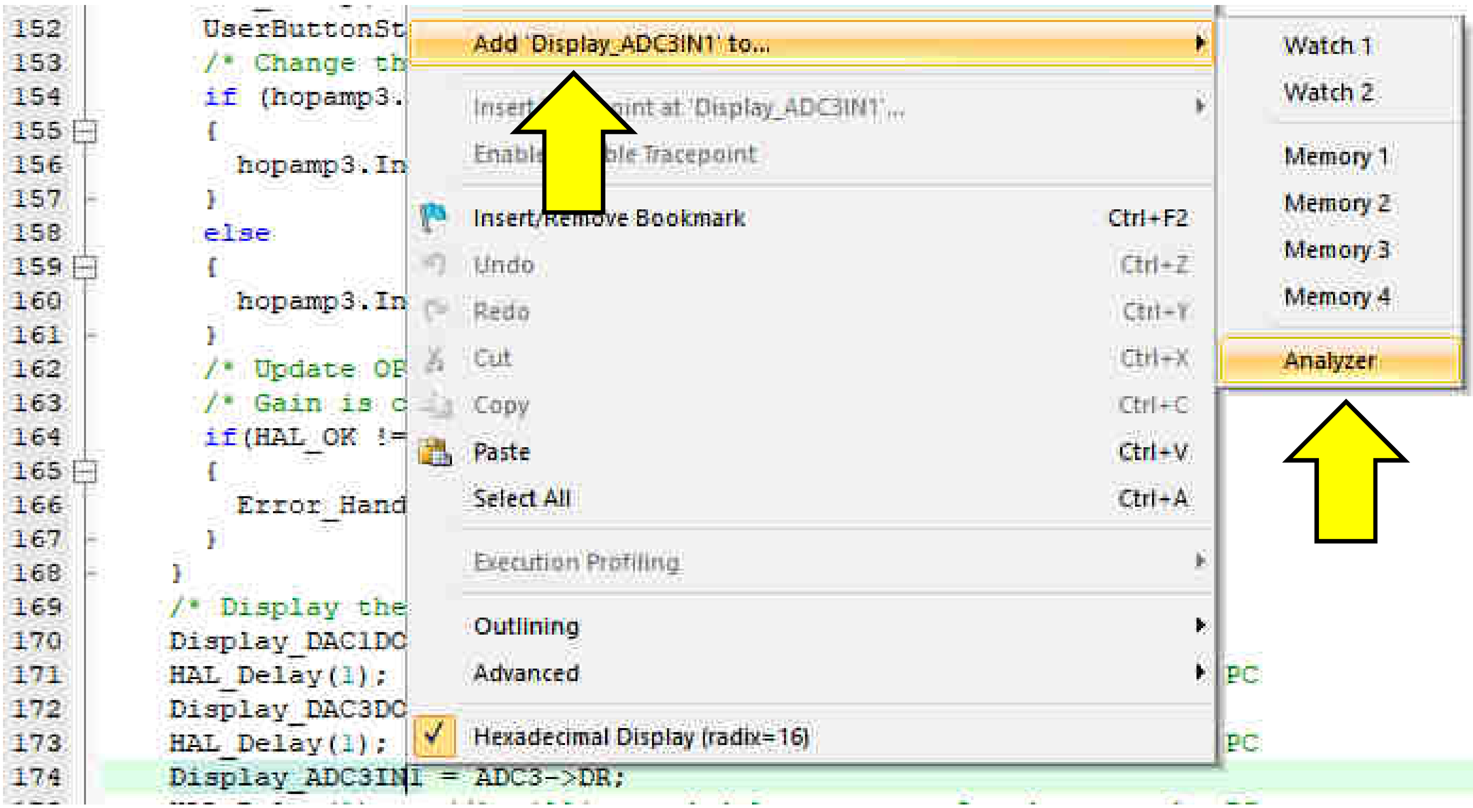
Load


C:\Users\jeffery blauser\STM32Cube\Repository\STM32Cube_FW_G4_V1.1.0\Projects\B-G47
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
LOAD
ADC_GainCompensation
Project
main.c

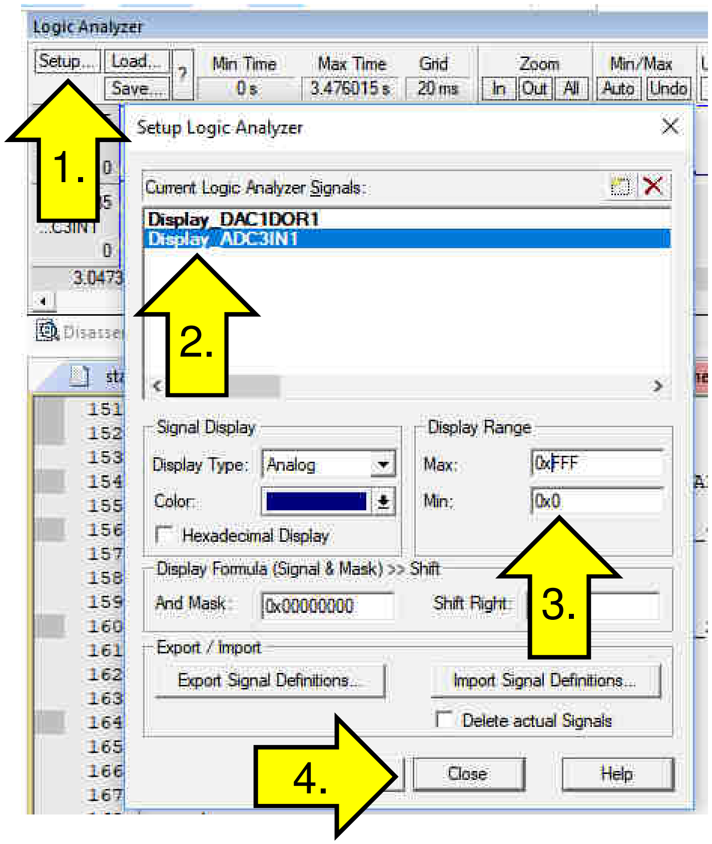
Start Debugger



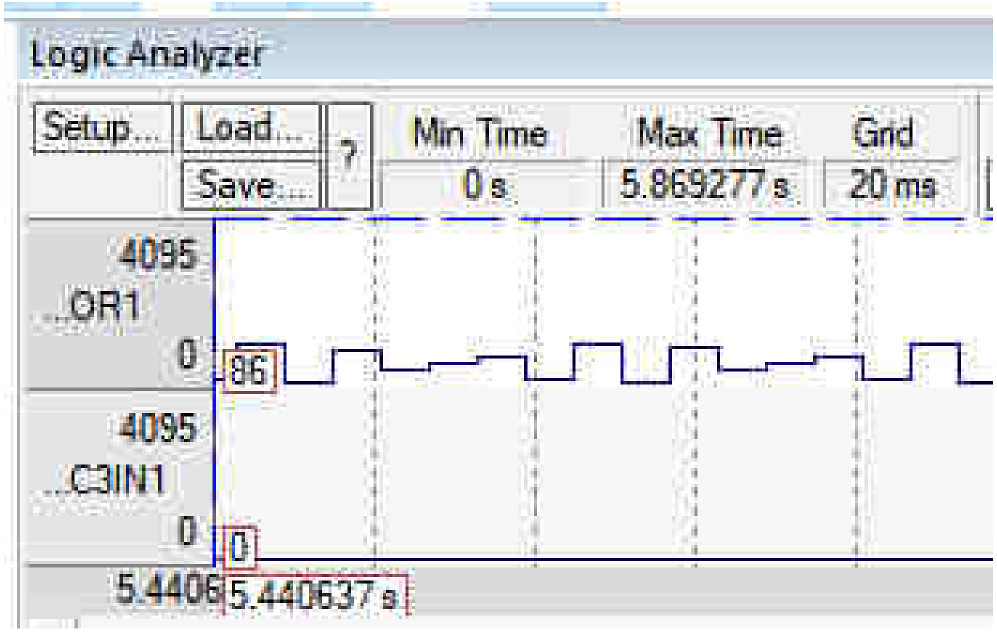
Add Display_ADC3IN1 to the Logic Analyzer



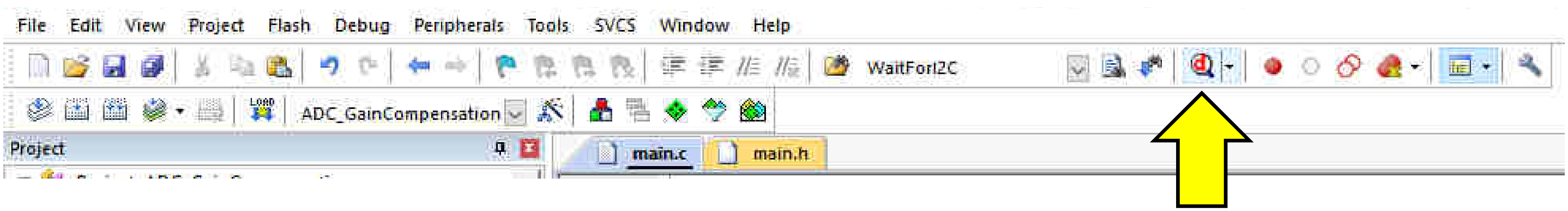
SETUP – ADC3IN1 – 0 to 0xFFF - Close



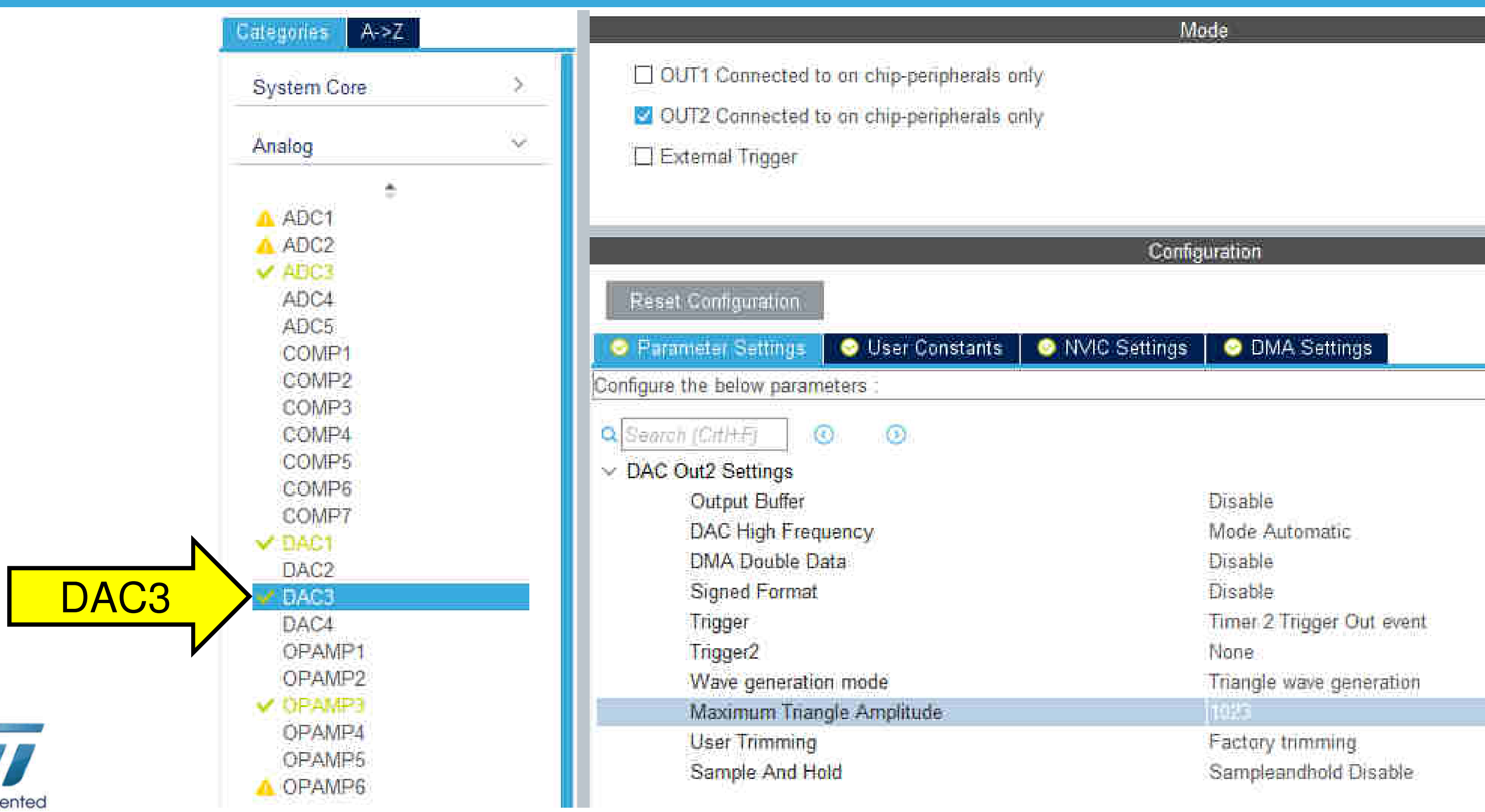
DAC output is good, Why is the ADC zero?



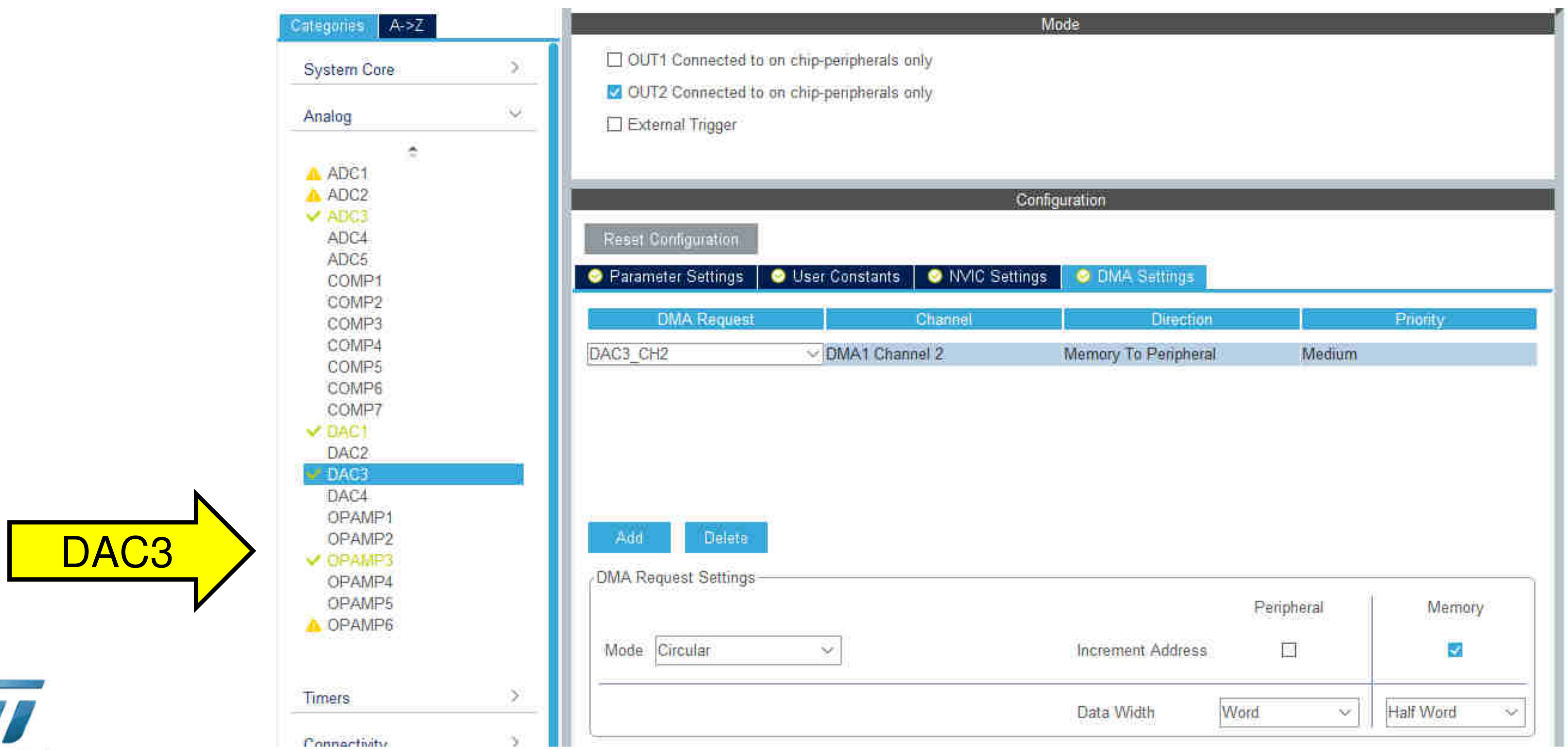
Exit Debugger



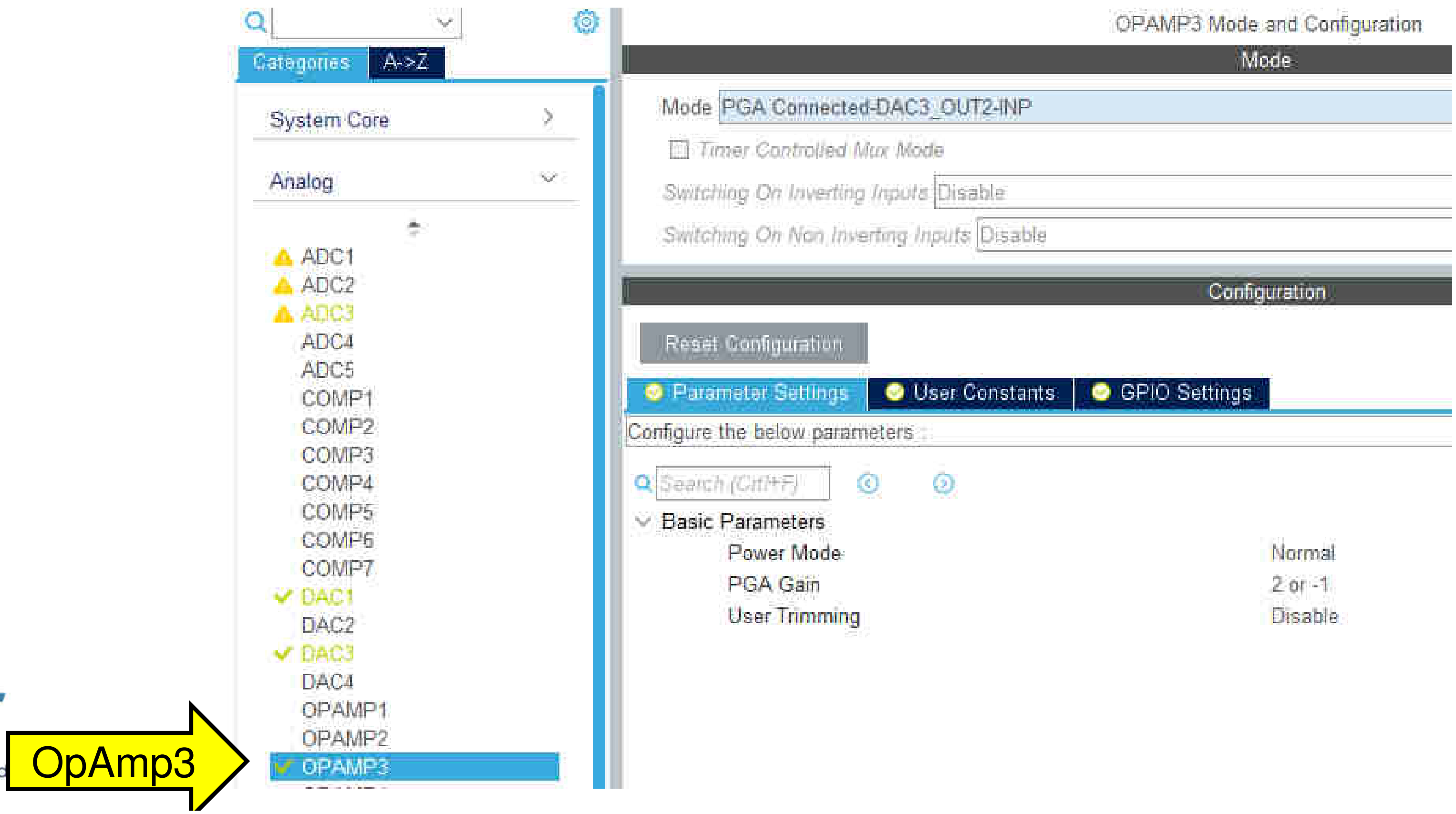
DAC3 – OUT2 – Timer2 Trigger Out Event



DAC3 – DMA - ADD - DMA1_CH2 – Periph=word / Mem=half word

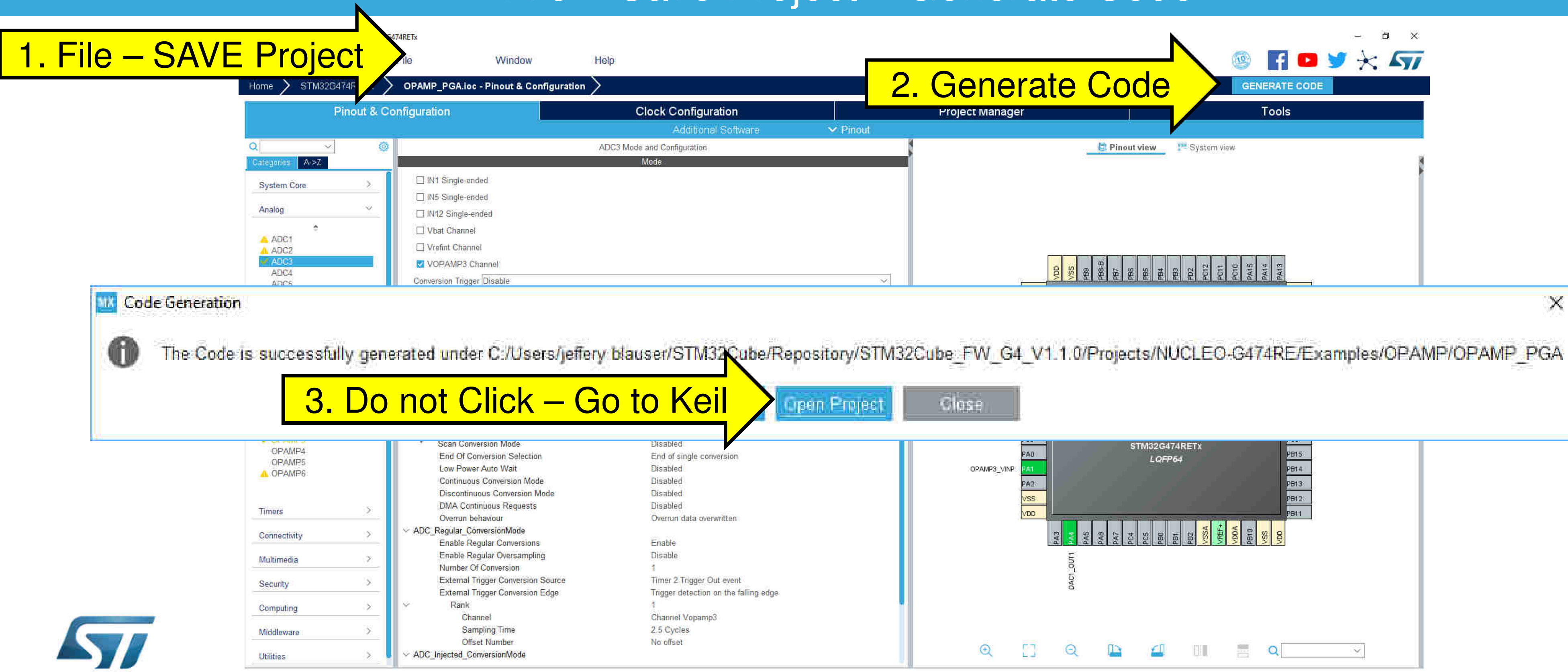


OpAmp3 Mode – PGA connected DAC3_OUT2-INP

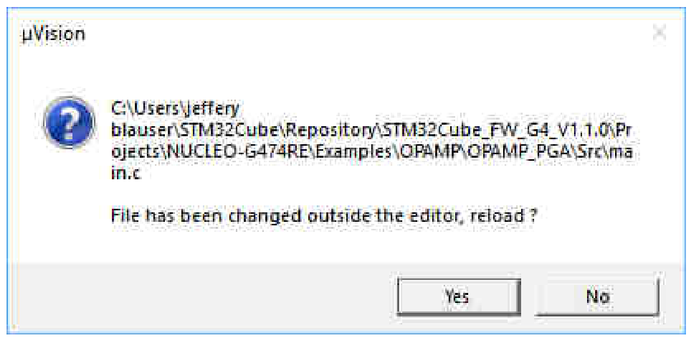


STM32CubeMx

File – Save Project – Generate Code



Click YES 3x


µVision
C:\Users\jeffery blauser\STM32Cube\Repository\STM32Cube_FW_G4_V1.1.0\Projects\NUCLEO-G474RE\Examples\OPAMP\OPAMP_PGA\Src\main.c
File has been changed outside the editor, reload ?
Yes
No

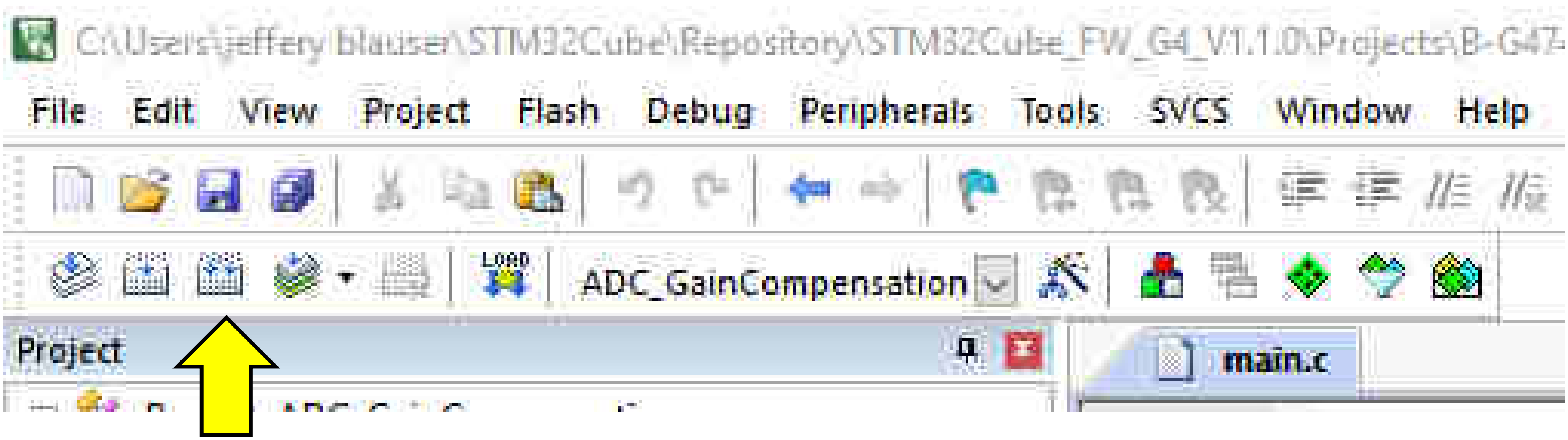
Keil MDK-ARM

In Main.c, change the direction of the DMA to DAC3_OUT2

```
/* Infinite loop */
/* USER CODE BEGIN WHILE */
/*##- Enable DAC Channel and associeted DMA ##############################*/
if(HAL_OK != HAL_DAC_Start_DMA(&hdac3, DAC_CHANNEL_2,
                               (uint32_t*)Sine12bit, 32, DAC_ALIGN_12B_R))
```

Keil MDK-ARM

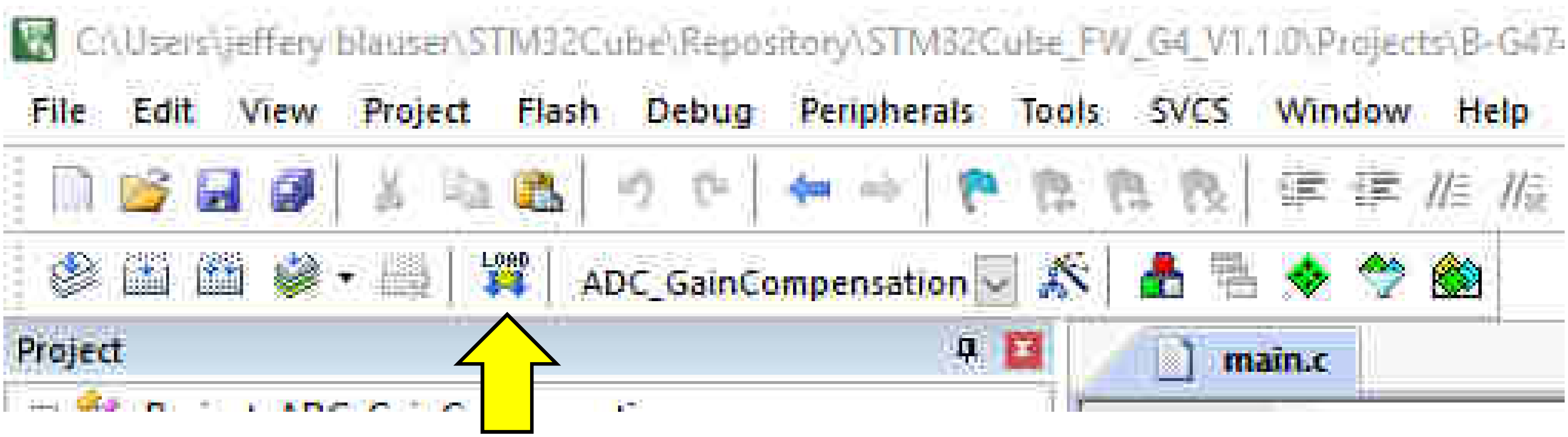
File-Save-Build



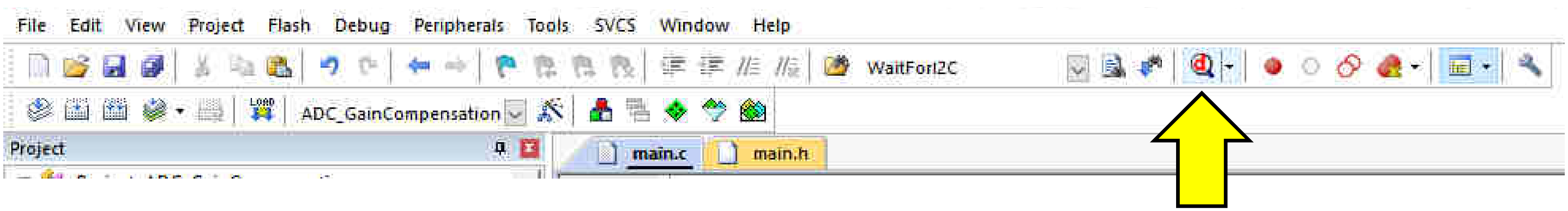
Build Output

```
Program Size: Code=6516 RO-data=596 RW-data=44 ZI-data=1268
FromELF: creating hex file...
OPAMP_PGA\OPAMP_PGA.hex: Warning: Q9931W: Your license for feature Keil will expire in 30 days
Finished: 0 information, 1 warning and 0 error messages.
"OPAMP_PGA\OPAMP_PGA.axf" - 0 Error(s), 0 Warning(s).
Build Time Elapsed:  00:01:22
```

Load

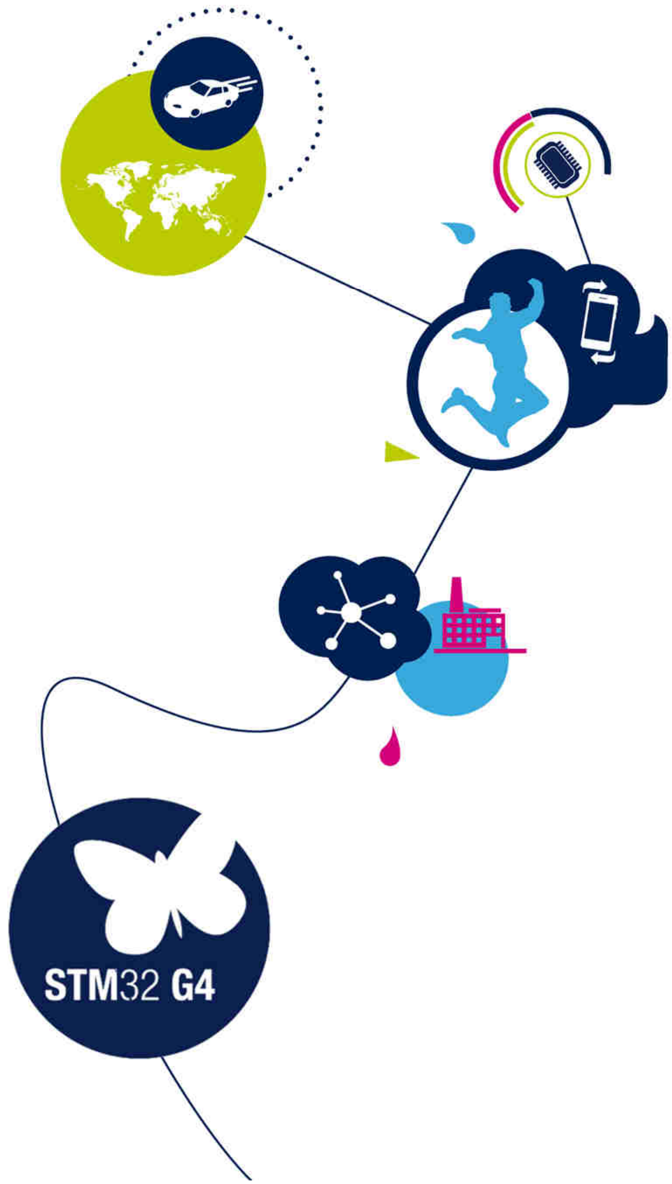

C:\Users\jeffery blauser\STM32Cube\Repository\STM32Cube_FW_G4_V1.1.0\Projects\B-G47
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
LOAD
ADC_GainCompensation
Project
main.c

Start Debugger



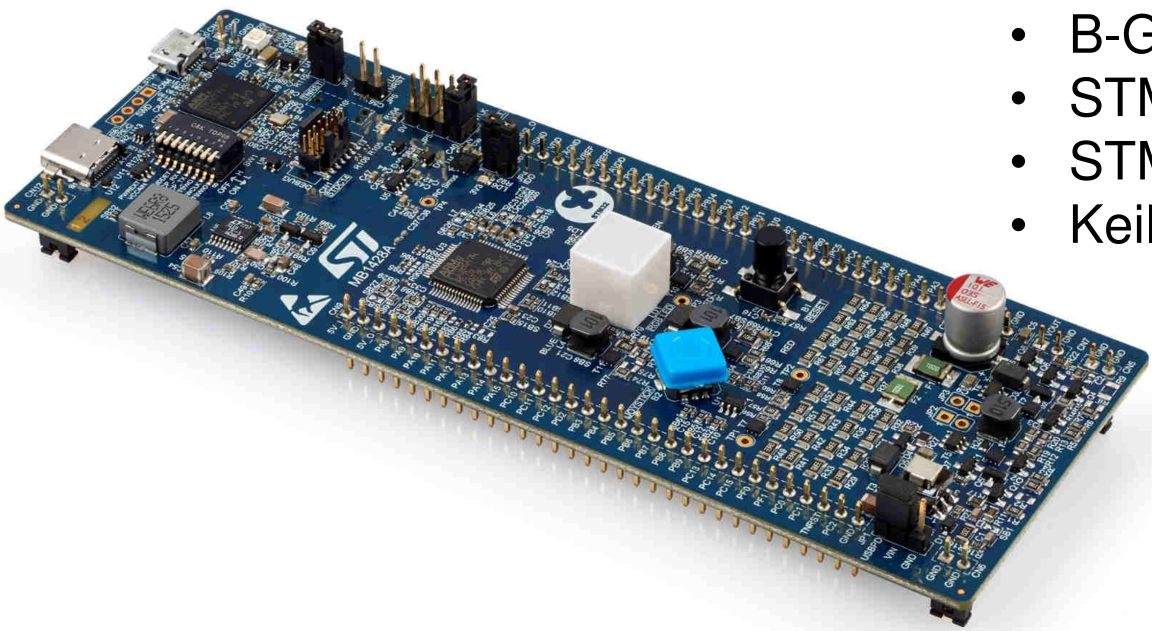
STM32G4 mixed-signal MCU workshop

Hands On:
High Resolution Timer (HRTIM)



life.augmented

Systems Check



- USB cable + USB-C Cable
- B-G474E-DPOW1 Discovery Board
- STM32G4G4 Family Cube Library
- STM32CubeMx
- Keil MDK-ARM

High Resolution Timer

G4 HRTIM

- Generates up to 12 outputs
- Up to 6 complementary pair PWM outputs
- 184pS resolution
- 6 sub timers, 4 compares each
- 32 set/reset sources for 12 outputs
- Sensitive to up to 10 external events
- Deadtime instertion down to 735pS
- Burst mode
- 8 interrupt vectors – 14 sources each

Internally connected to:

- DMA
- ADC's
- DAC's
- Comparators
- Timers
- NVIC

G4 HRTIM – What's new?

The timer F unit has been added to have a total of 12 outputs available, together with a 6th FAULT input

Rational: possibility to address larger converters

- Triple-interleaved half-bridge LLC, requiring 3 x 4 outputs
  - 2x at primary for half-bridge control and 2x at secondary for synchronous rectification
- Dual interleaved full-bridge LLC (2x 6 outputs)
- Dual phase-shift full bridge converter (2x 6 outputs)

Implementation

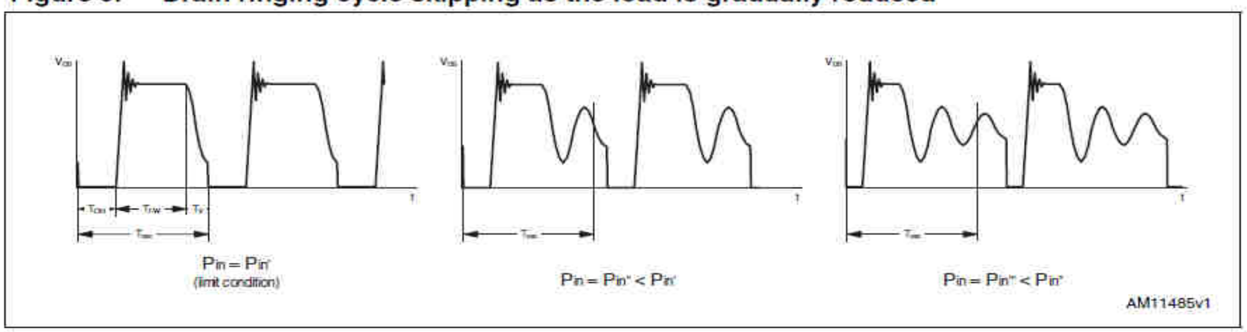
- Timer F new control / status bits have been added in reserved areas, to maintain partial compatibility whenever possible
- Still, as many registers were full, the option was chosen to maintain all bits in single 32-bit registers, but breaking compatibility
- When bit position was changed, this is managed transparently by the HAL / LL

G4 HRTIM – What's new? – External Event counter 1 of 2

- The HRTIM v1 offers many options for timing-driven filtering (blanking and windowing)
  - Typical use case: leading edge blanking (LEB)

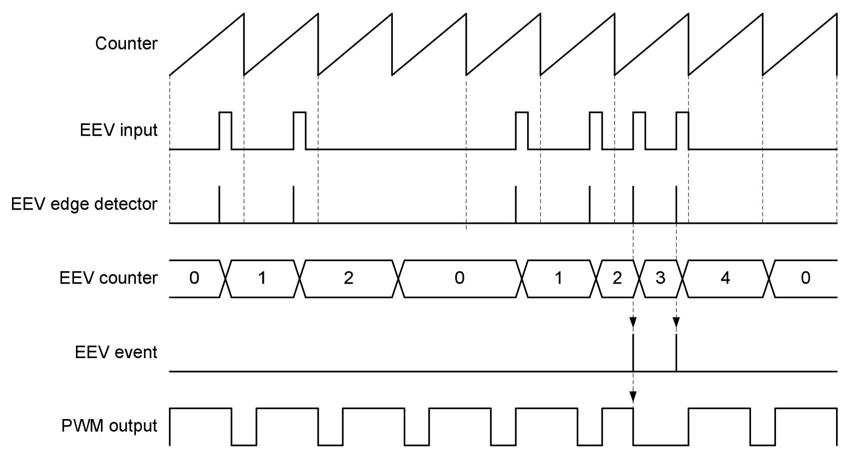
- The HRTIM v2 features event-based filtering: an external event must have occurred a given number of times before being considered valid
  - Typical use case: valley skipping mode for flyback converters
    - At low-load, the next PWM cycle must start after N cycles of ringing (the lower the load, the higher N)

Figure 9. Drain ringing cycle skipping as the load is gradually reduced



G4 HRTIM – What's new? – External Event counter 2 of 2

- Immediate mode
  - Event is generated as soon as N consecutive events are occurring during a PWM period
    - Event counter is reset on each and every new PWM period
- Cumulative mode (figure below)
  - event must occur at least once during multiple PWM periods
    - counter is reset only if the event did not appear during the last PWM period



High Resolution Timer

Lab steps 1 of 4

- This Hands On example shows how to configure the HRTIM to control a non-inverting buck-boost converter timer.
- This example uses the STM32G4xx HRTIM HAL and LL APIs.
- You will use the timer unit C and D and TC1/TC2/TD1/TD2 outputs (respectively PB12, PB13, PB14, PB15).

High Resolution Timer

Lab steps 2 of 4

- The FAULT2 input is enabled on PA15 (active LO) to demonstrate PWM shut down (low level sensitive), for all outputs.
- When the fault is triggered (PA15 input connected to GND), TC1, TC2, TD1, TD2 signals are shut-down. The system can be re-armed by pressing the user button.
- ADC1 is configured to have conversions triggered in the middle of the converter ON time, on PA1 (BUCKBOOST_VIN) and PA3 (BUCKBOOST_VOUT) inputs.

life.augmented

Lab steps 3 of 4

- To run the demo, the Vin input pin of the BUCK-BOOST converter must be connected to the 5V_O supply on the CN9 connector or on an external power supply. The resulting voltage is available on Vout pin

- LEDs are indicating the following:
  - Green LED4: blinks during BUCK operation
  - Orange LED3: blinks during BOOST operation
  - Red LED5: blinks when FAULT is triggered

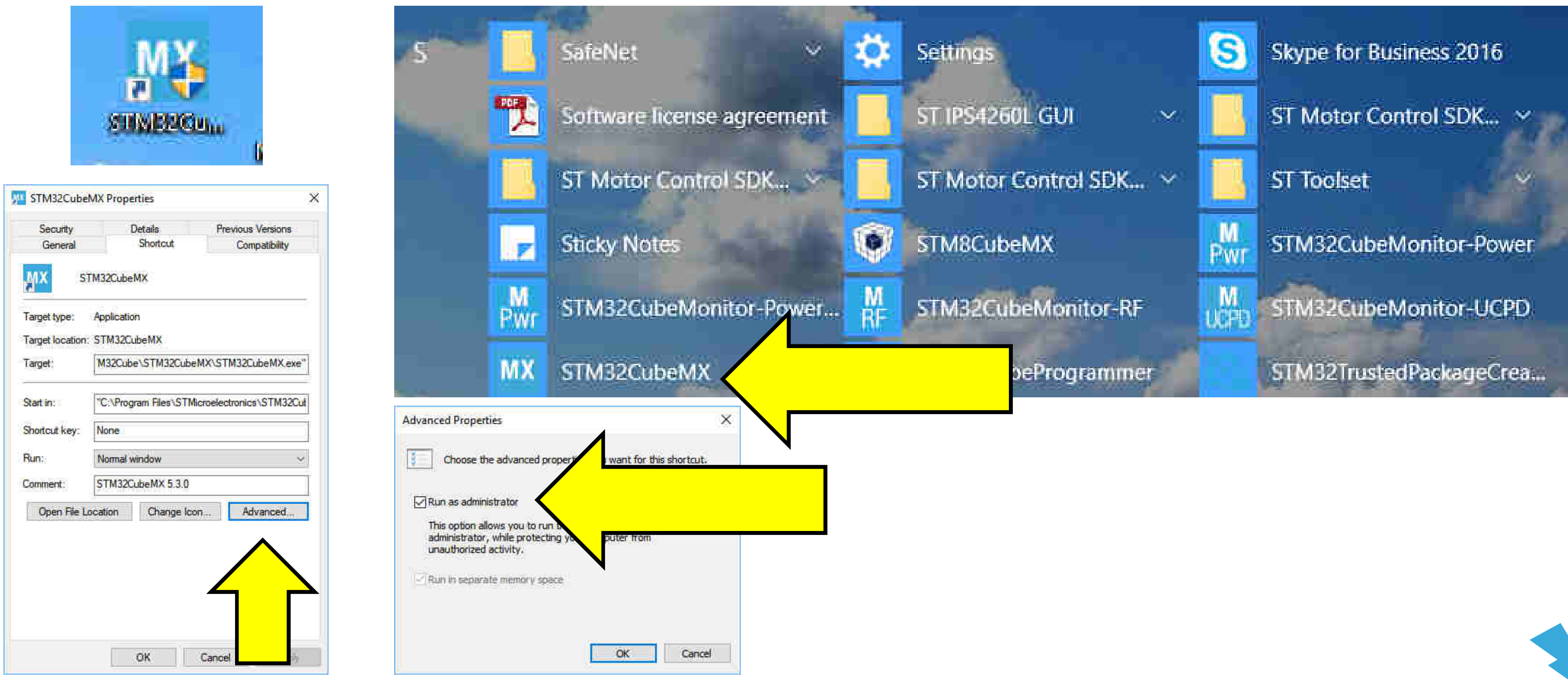
High Resolution Timer

Lab steps 4 of 4

- The demo starts in BUCK mode, and the duty cycle is slowly adjusted in the TIMC IRQ handler to have Vout continuously varying below Vin value.
- If the push-button is pressed and the voltage is below 5V, the boost mode is enabled (this is to prevent exceeding the Discovery Kit max output voltage (15V).
- The voltage is increased above Vin value (with a fixed duty cycle).
- If the push-button is pressed again, the output capacitor is first de-energized down to 2.5V before re-enabling the buck mode.

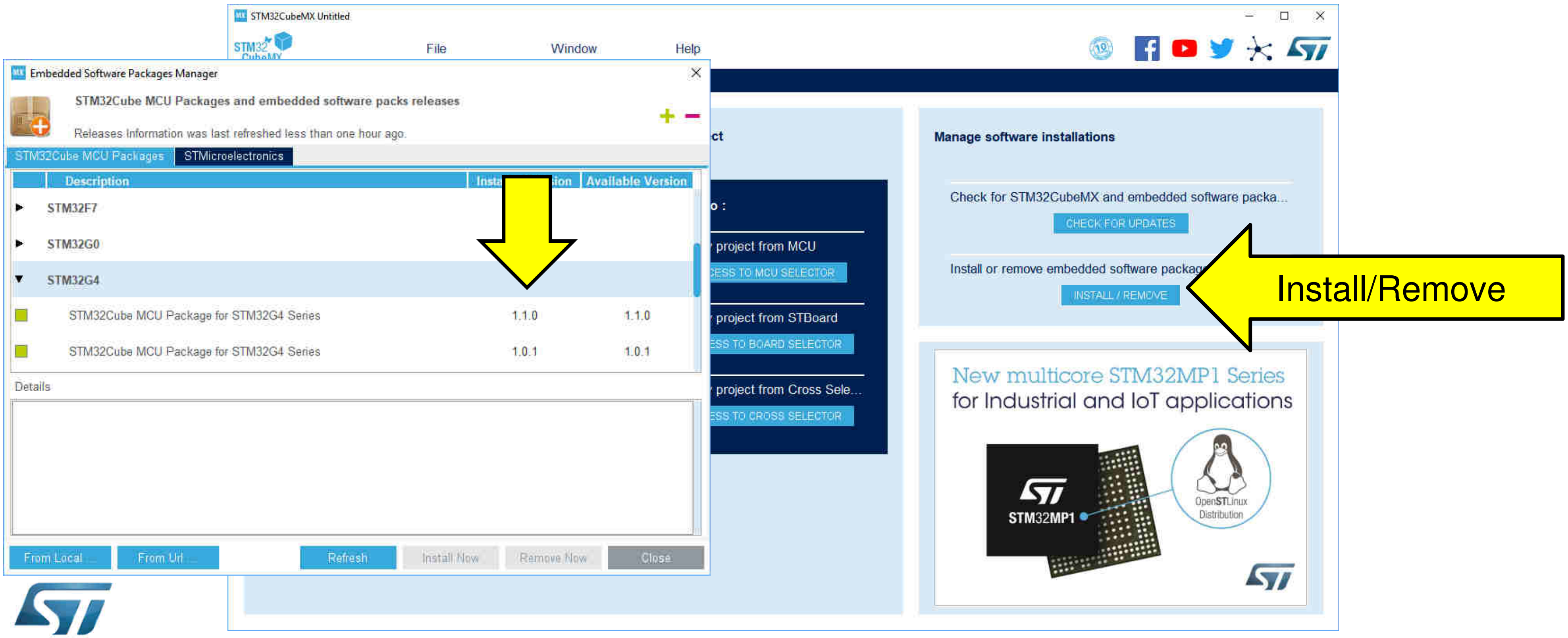
STM32CubeMx

Start up



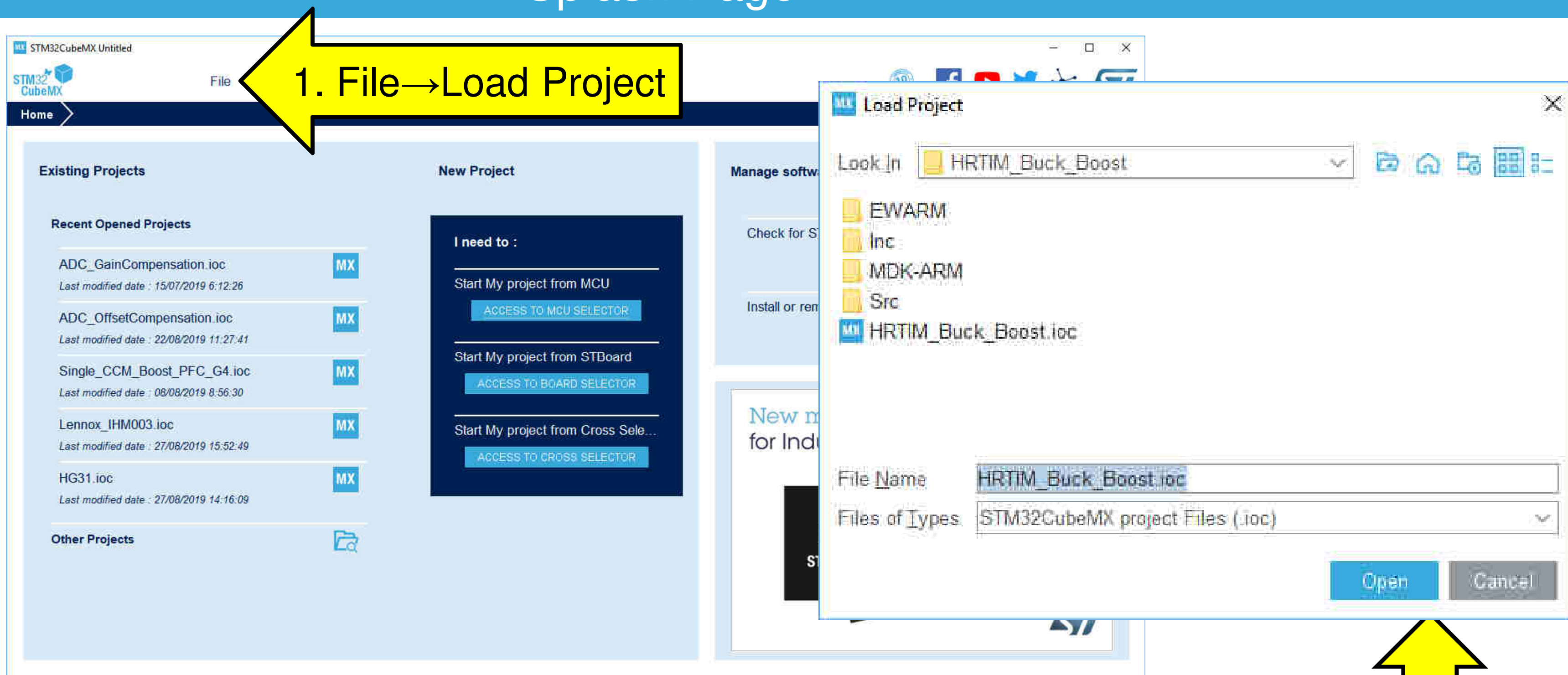
STM32CubeMx

Splash Page – G4 Library Installed?



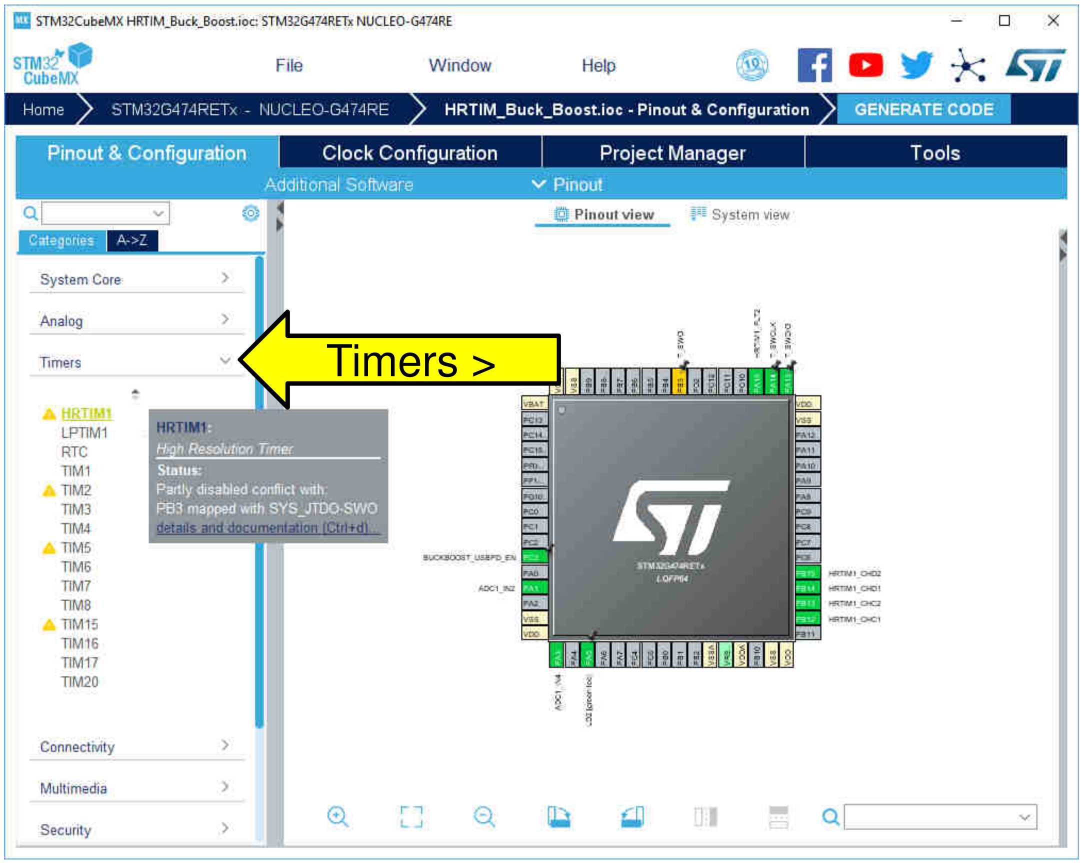
STM32CubeMx

Splash Page

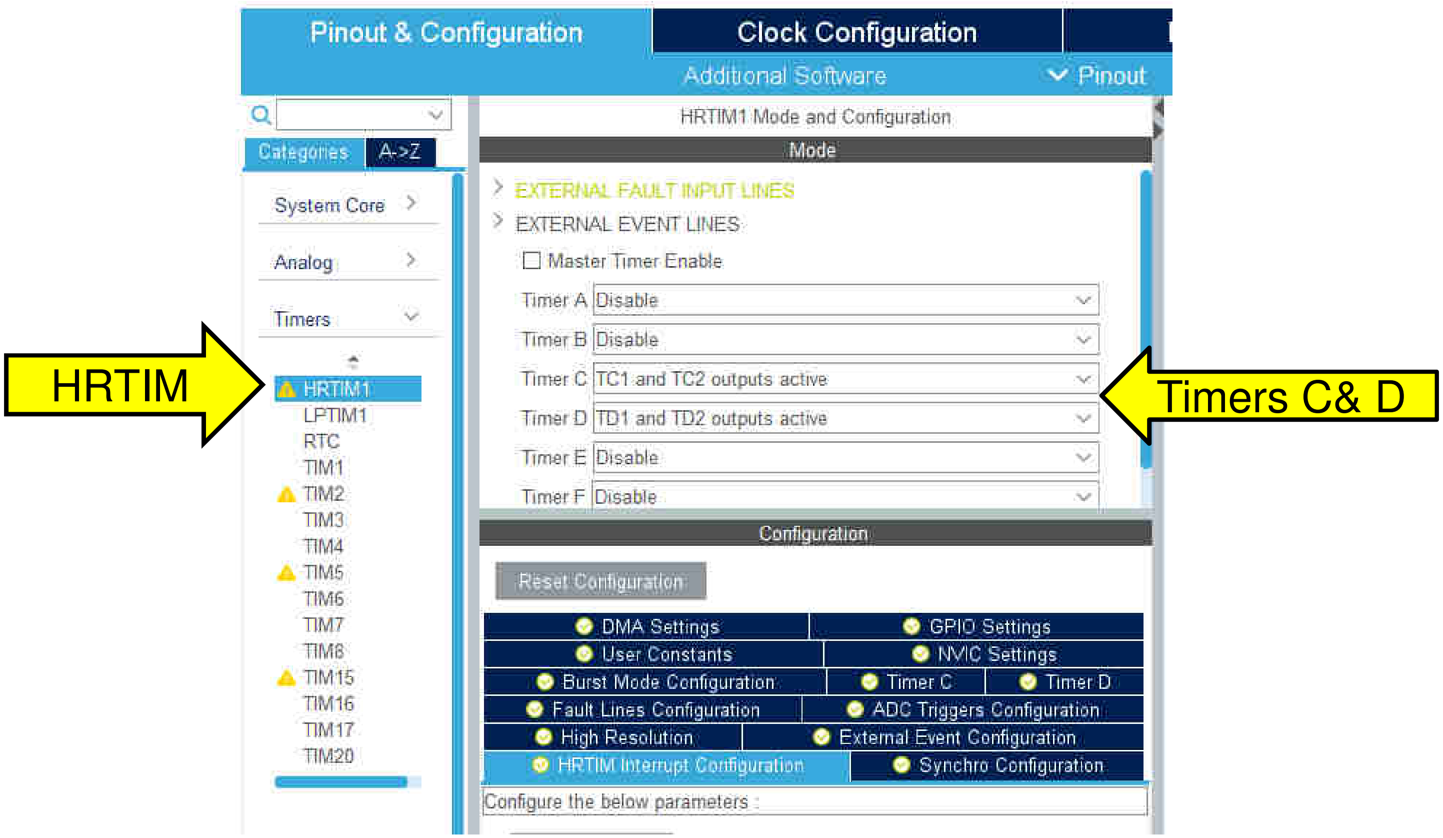


C:\...\STM32Cube\Repository\STM32Cube_FW_G4_V1.1.0\Projects\B-G474E-DPOW1\Examples_MIX\HRTIM\HRTIM_Buck_Boost

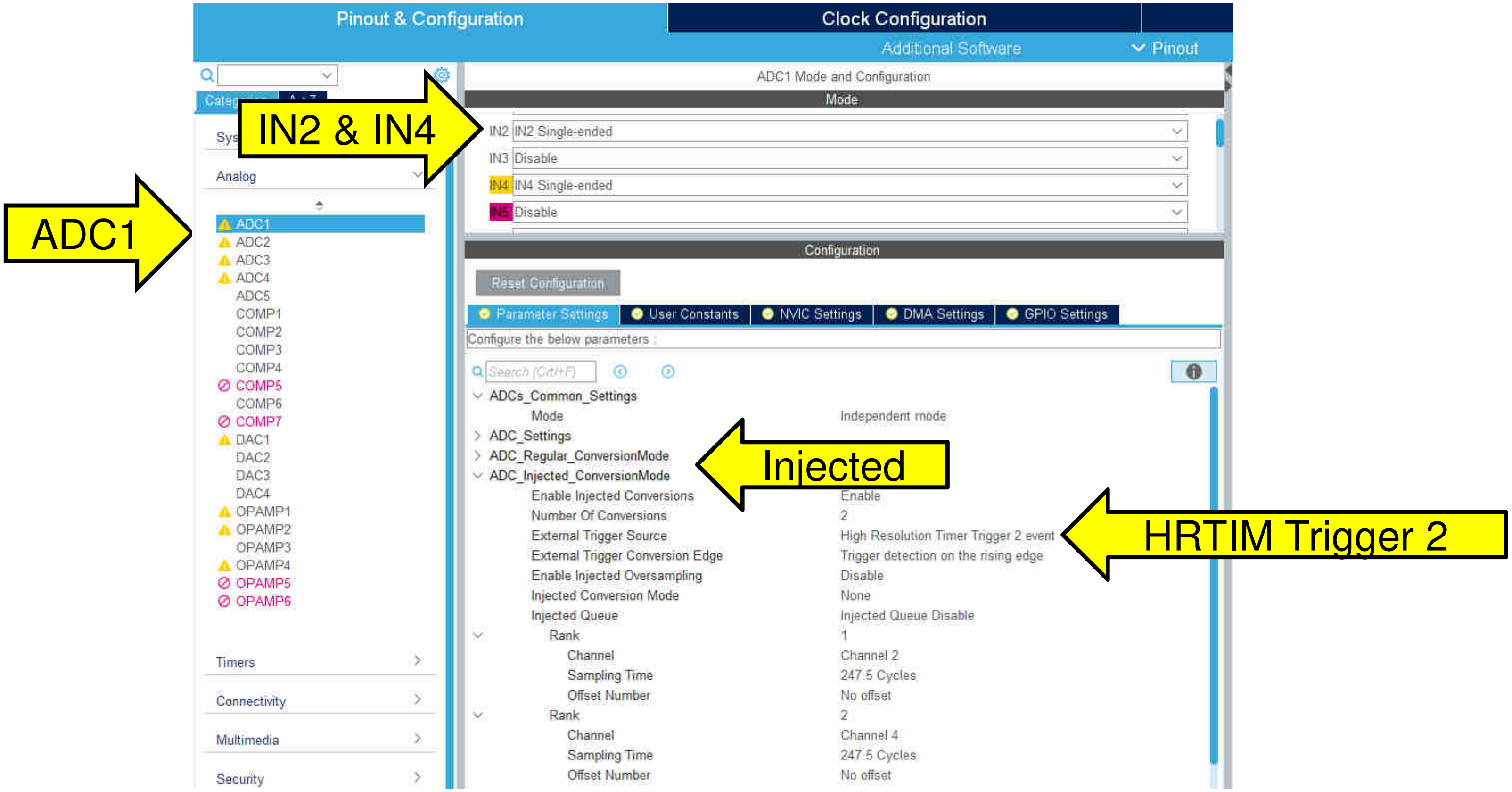
Graphic Design Representation



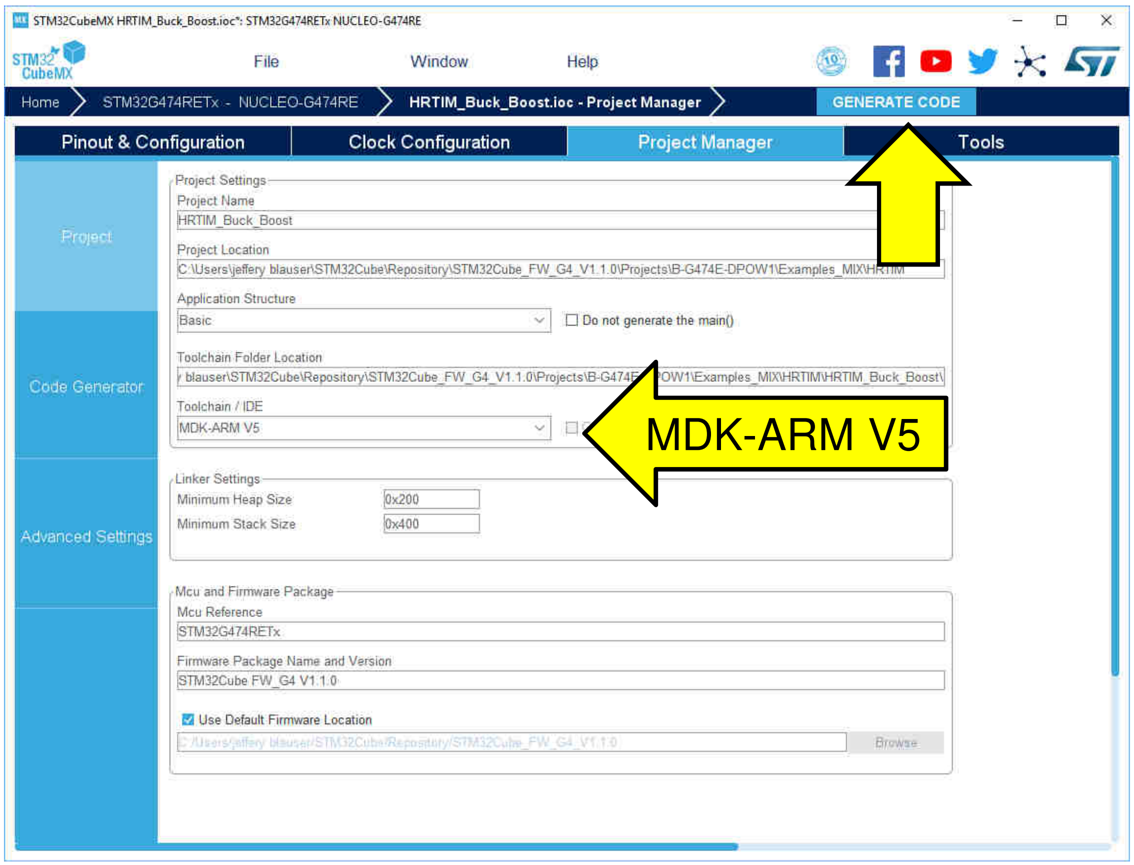
HRTIM Channels C & D



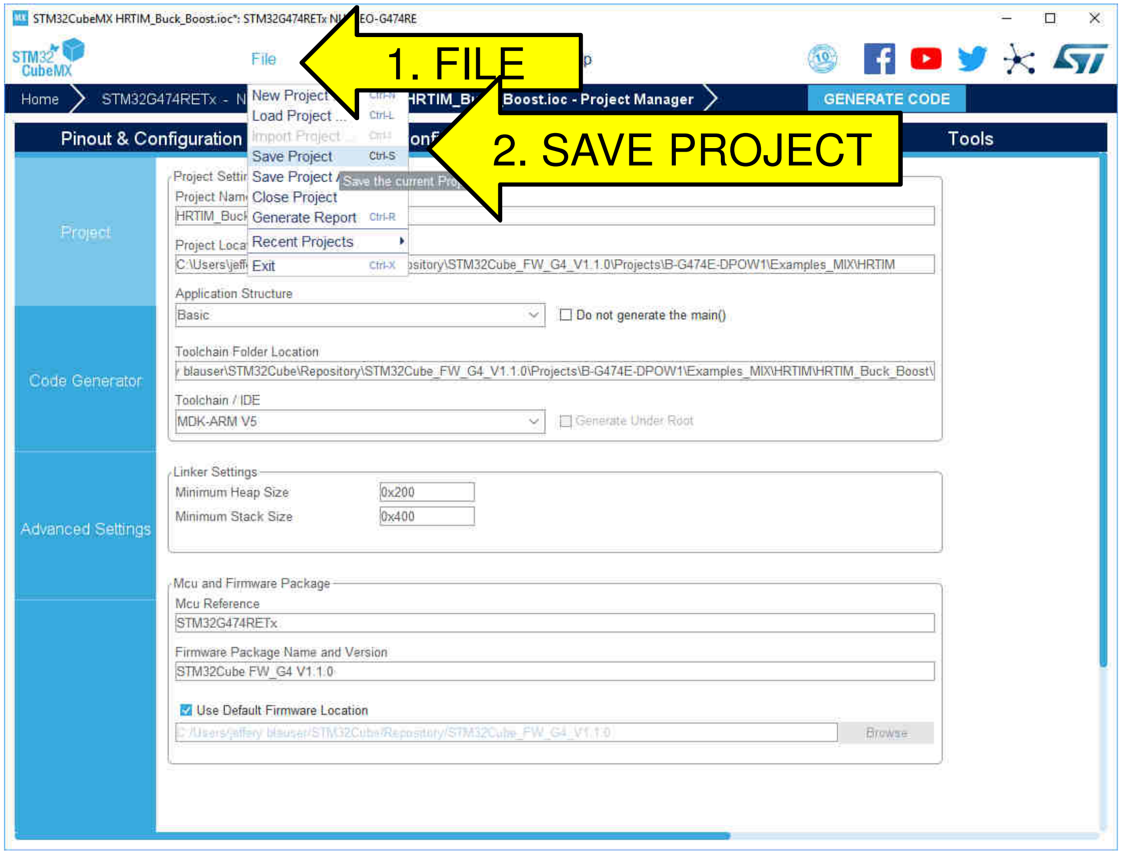
HRTIM – ADC – 2 injected input channels



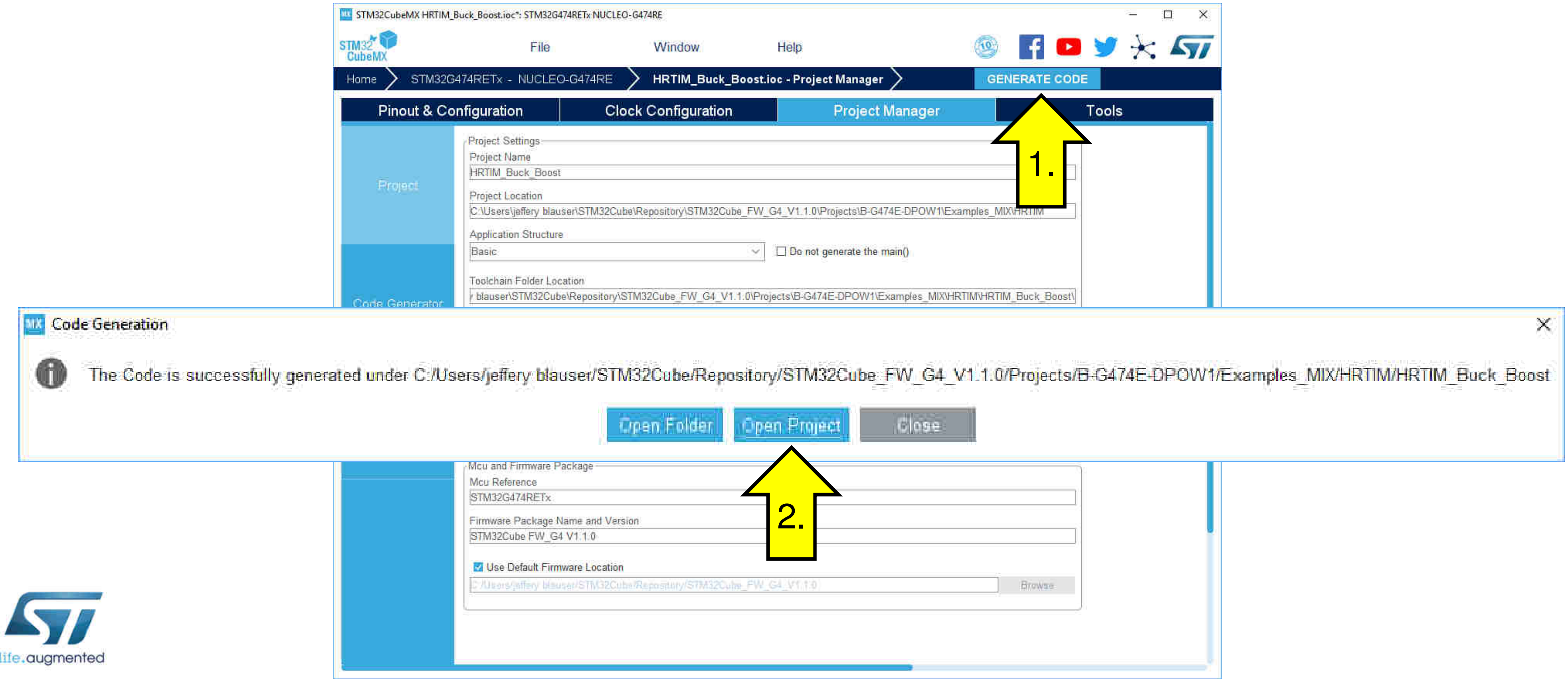
Project Manager – Keil MDK-ARM V5



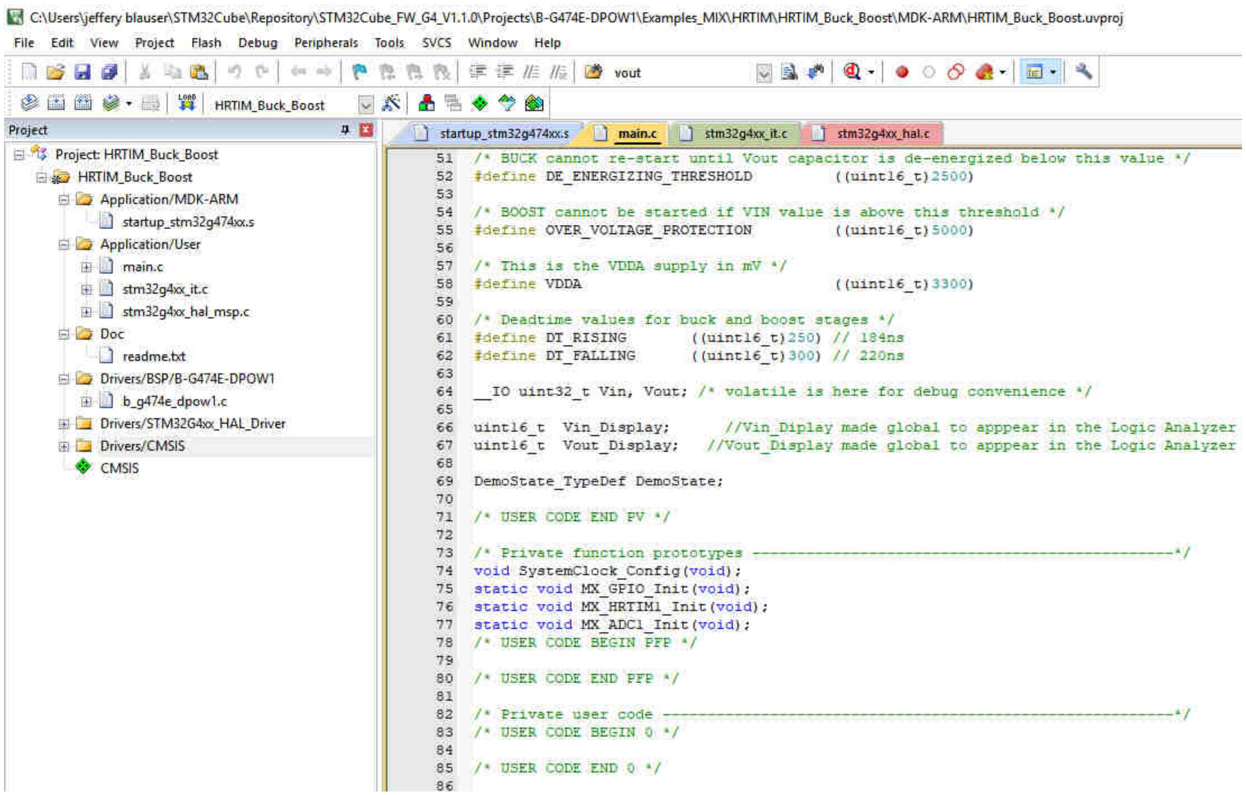
FILE→SAVE PROJECT



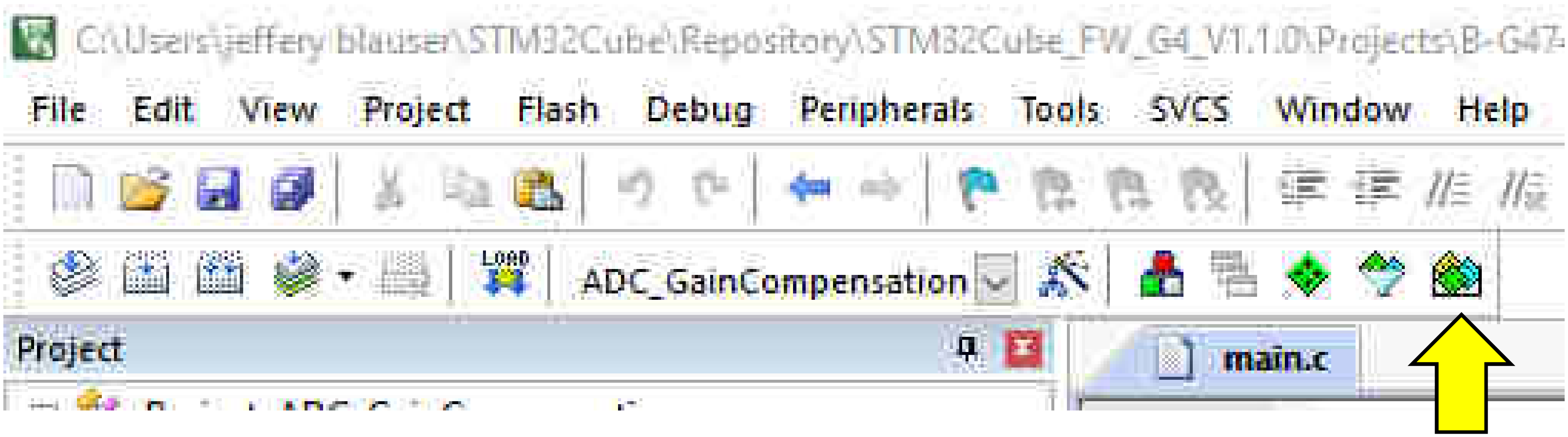
Generate Code



Keil Start up page

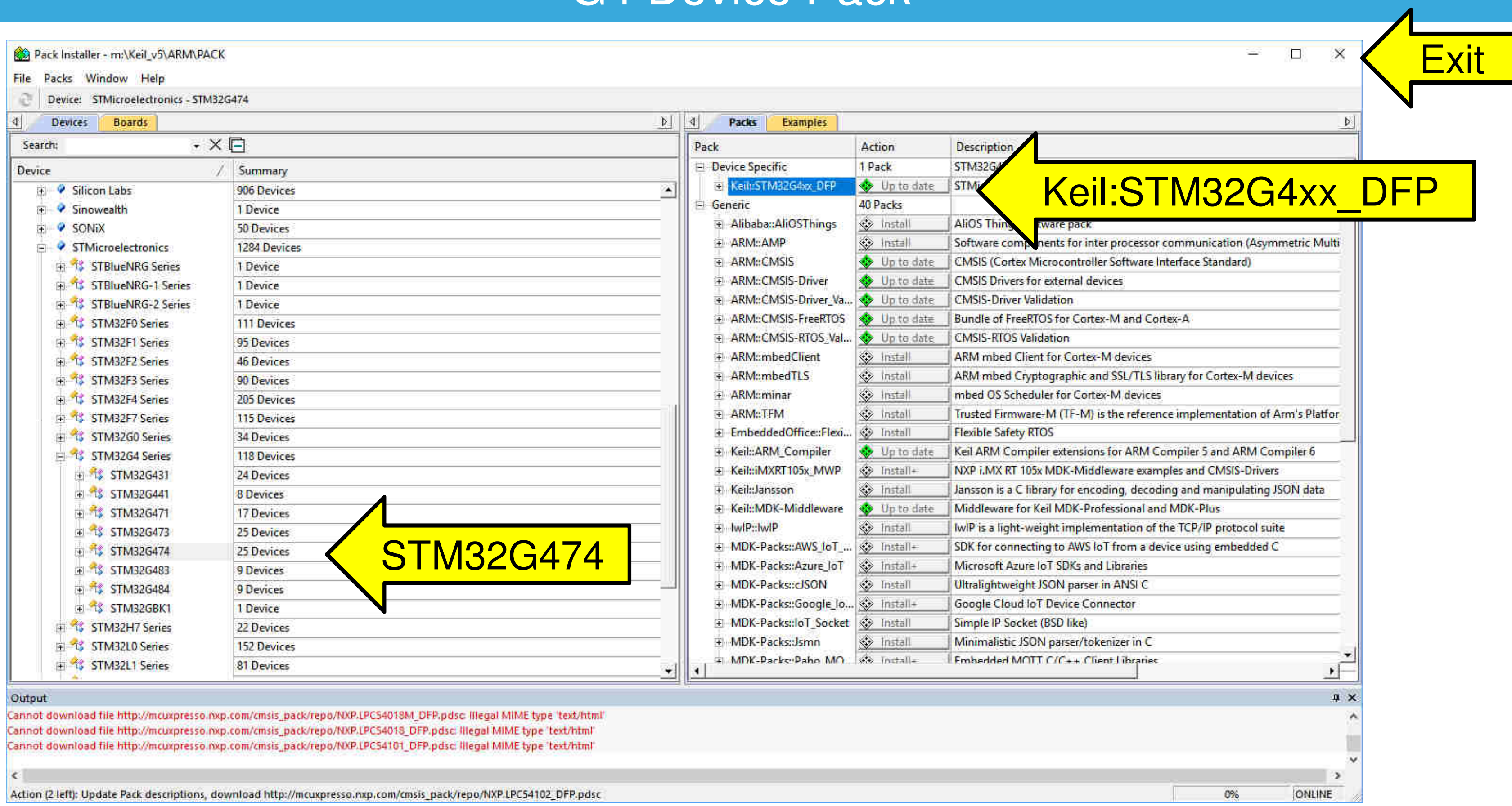


Check the Device Pack



Keil MDK-ARM

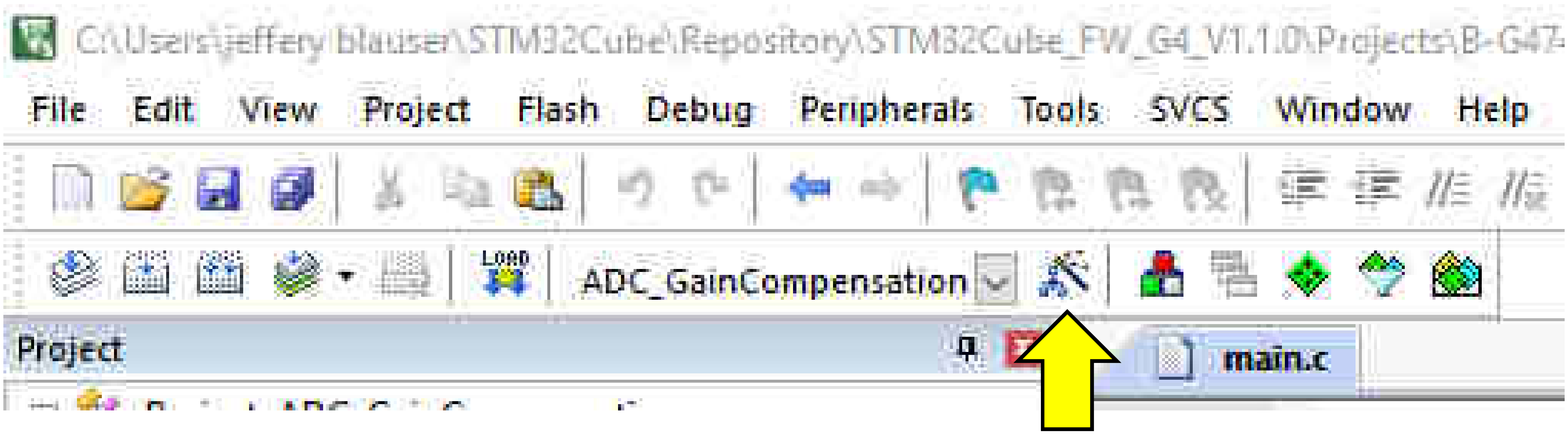
G4 Device Pack



life.augmented

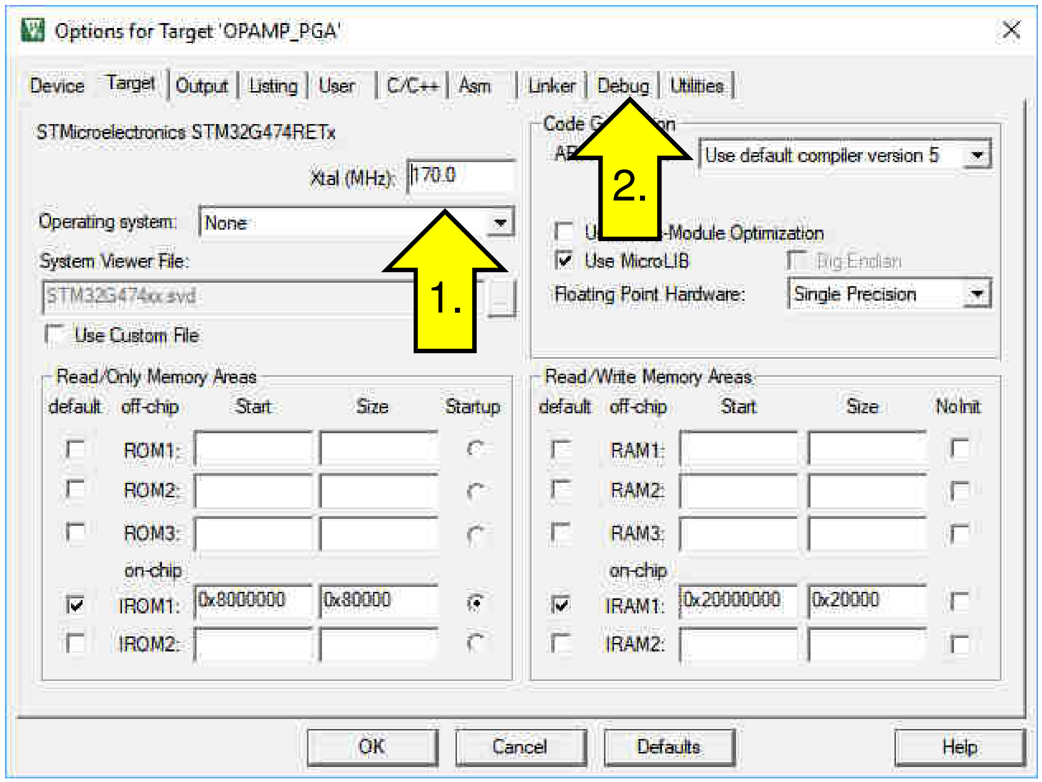
Keil MDK-ARM

Check the Options

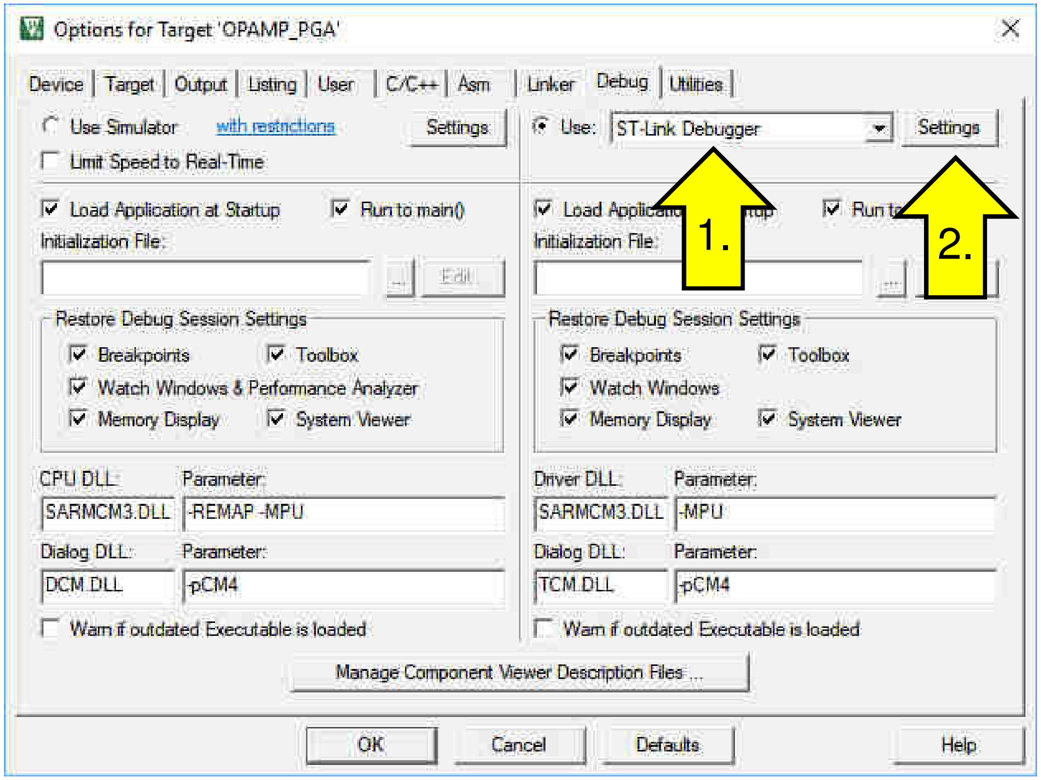


Keil MDK-ARM

Clock speed and Debug

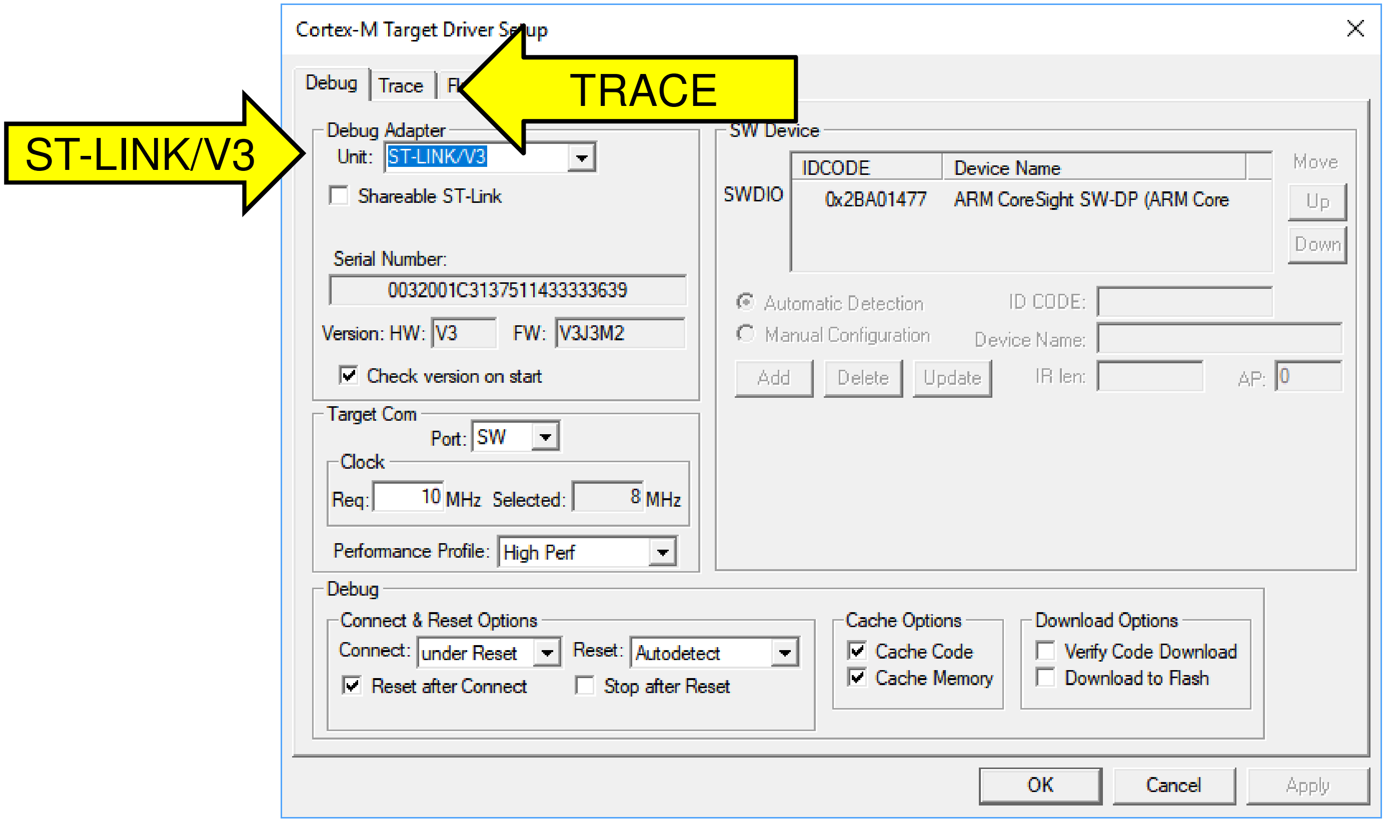


ST-Link Debugger - Settings



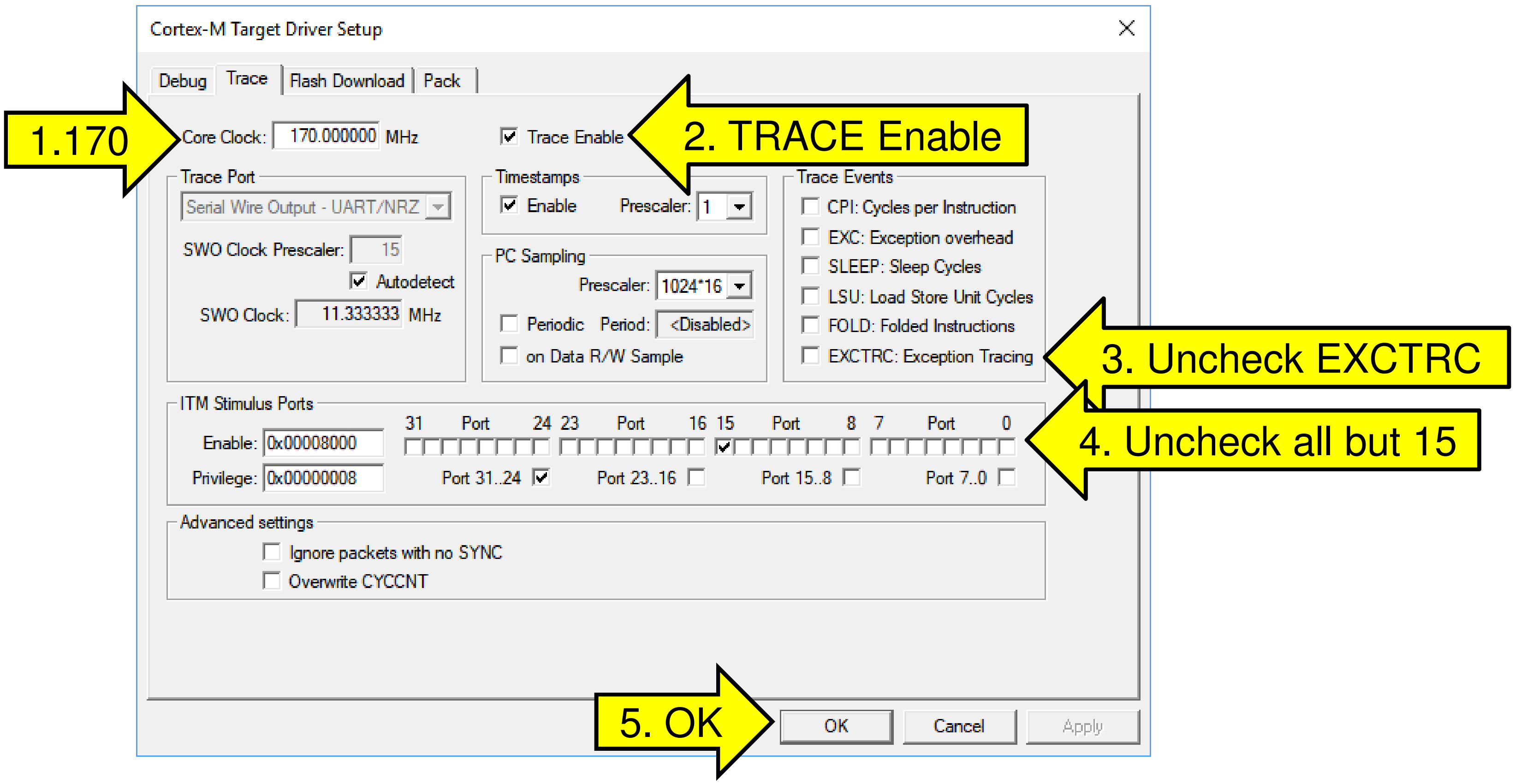
Keil MDK-ARM

ST-Link/V3 & TRACE



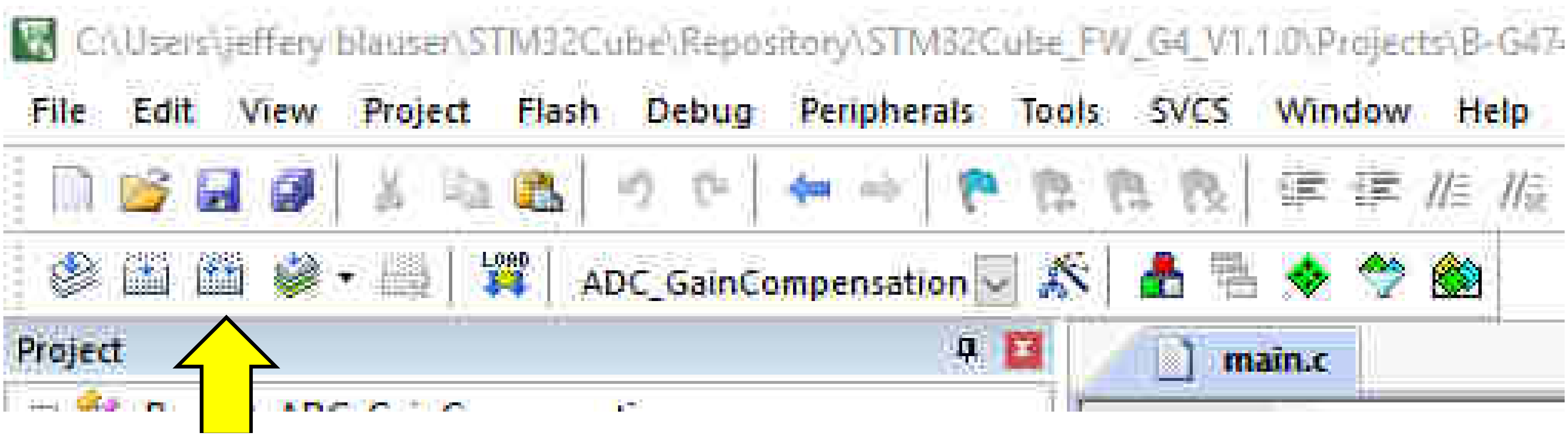
Keil MDK-ARM

Core Clock & TRACE Enable (After)



Keil MDK-ARM

Build

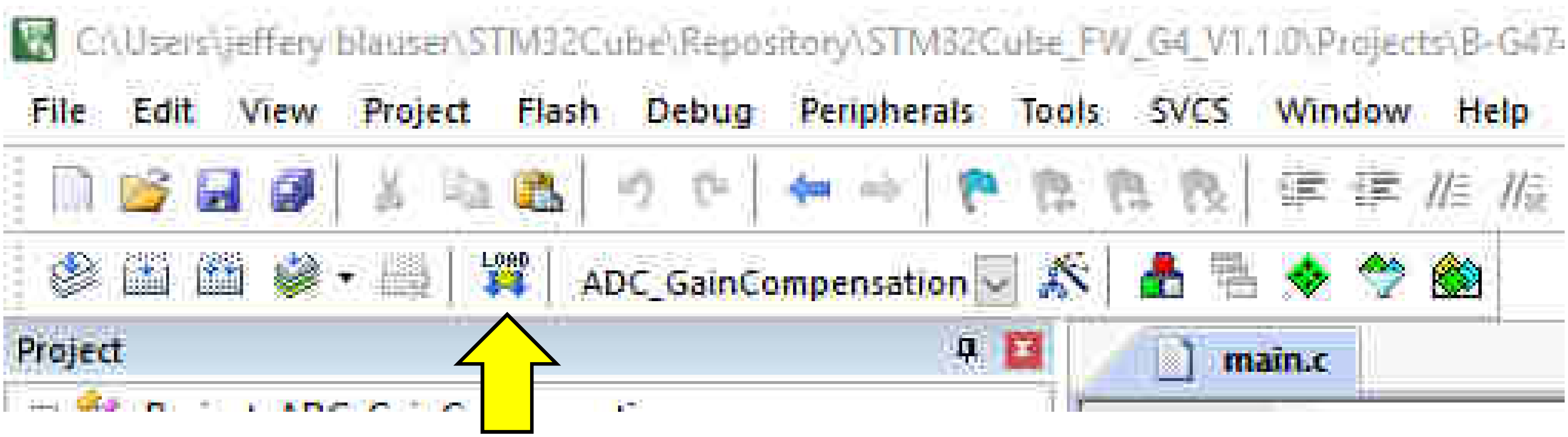


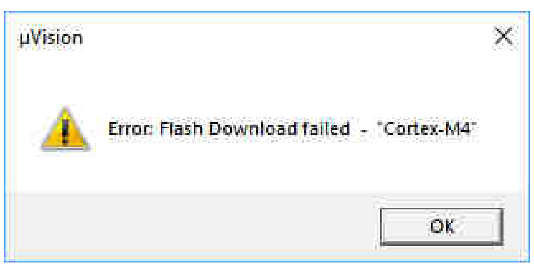
Build Output

```
Program Size: Code=15028 RO-data=604 RW-data=92 ZI-data=1428
FromELF: creating hex file...
HRTIM_Buck_Boost\HRTIM_Buck_Boost.hex: Warning: Q9931W: Your license for feature Keil will expire in 29 days
Finished: 0 information, 1 warning and 0 error messages.
"HRTIM_Buck_Boost\HRTIM_Buck_Boost.axf" - 0 Error(s), 0 Warning(s).
Build Time Elapsed:  00:01:33
```

Keil MDK-ARM

Load



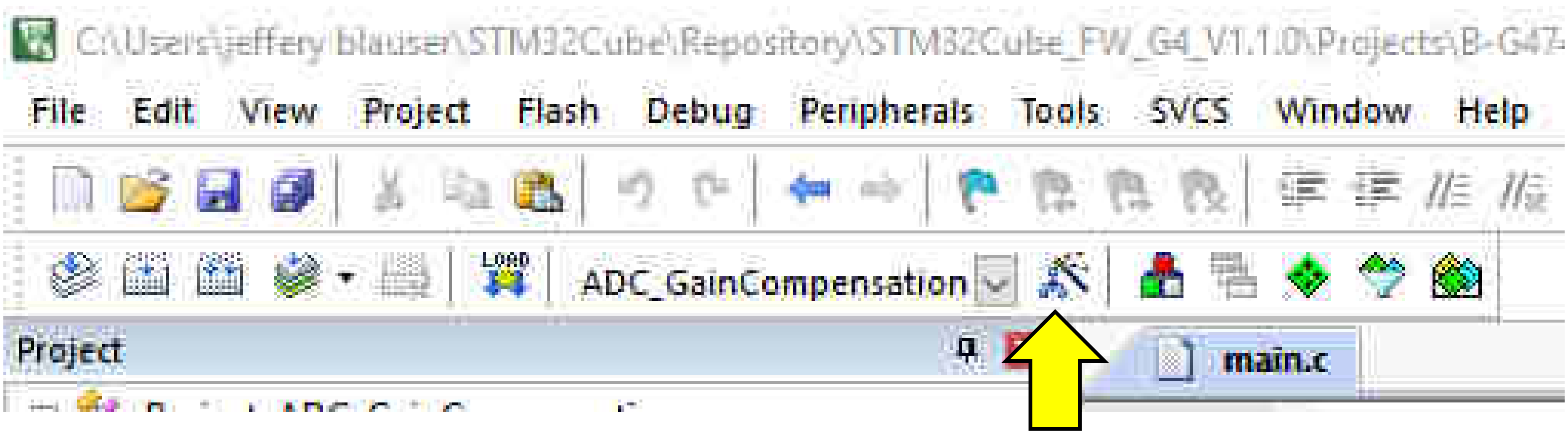


Build Output

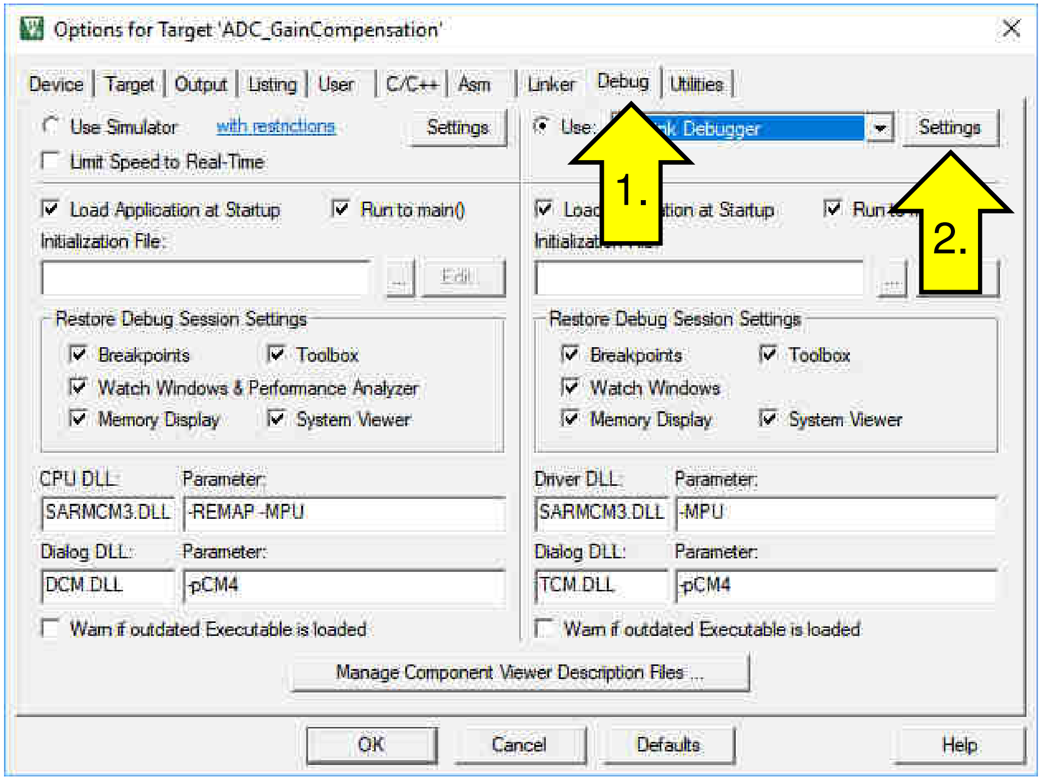
```
Contents mismatch at: 08001860H  (Flash=38H  Required=C5H) !
Contents mismatch at: 08001861H  (Flash=46H  Required=F8H) !
Contents mismatch at: 08001862H  (Flash=FEH  Required=88H) !
Contents mismatch at: 08001863H  (Flash=F7H  Required=00H) !
Contents mismatch at: 08001864H  (Flash=B9H  Required=60H) !
Contents mismatch at: 08001865H  (Flash=FCH  Required=6CH) !
Too many errors to display !
Error: Flash Download failed  -  "Cortex-M4"
```

Keil MDK-ARM

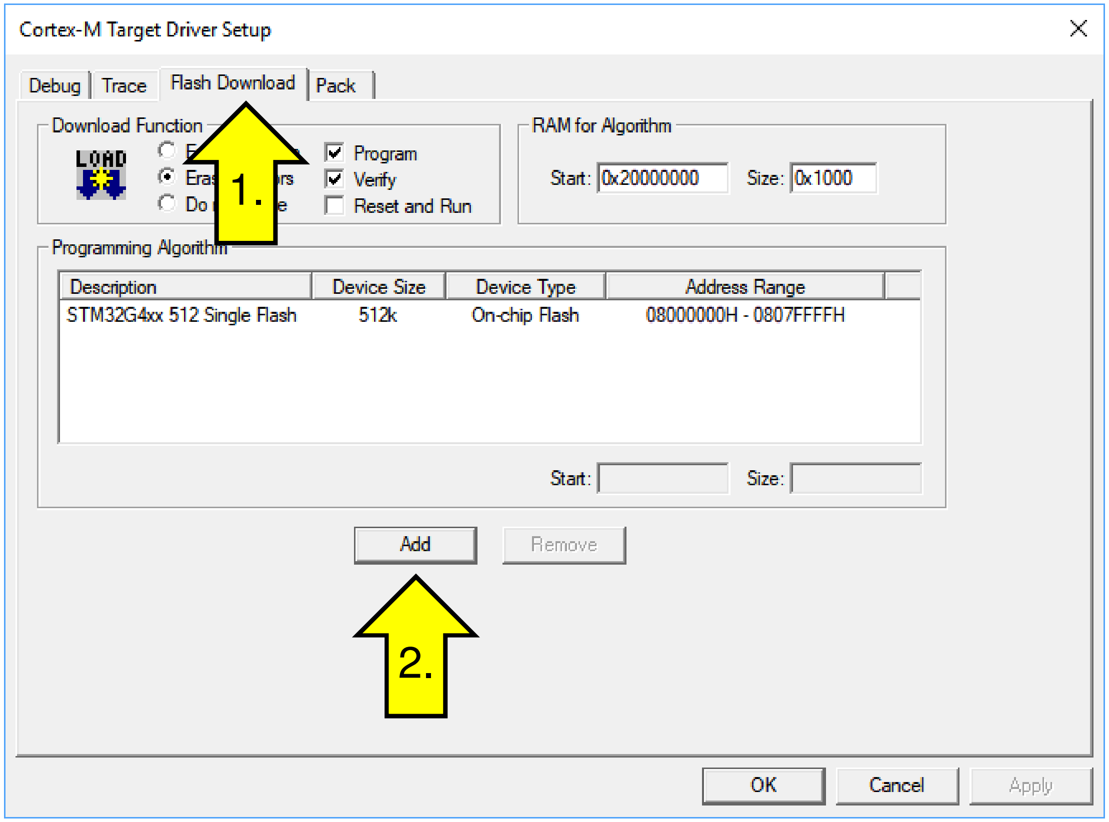
Options



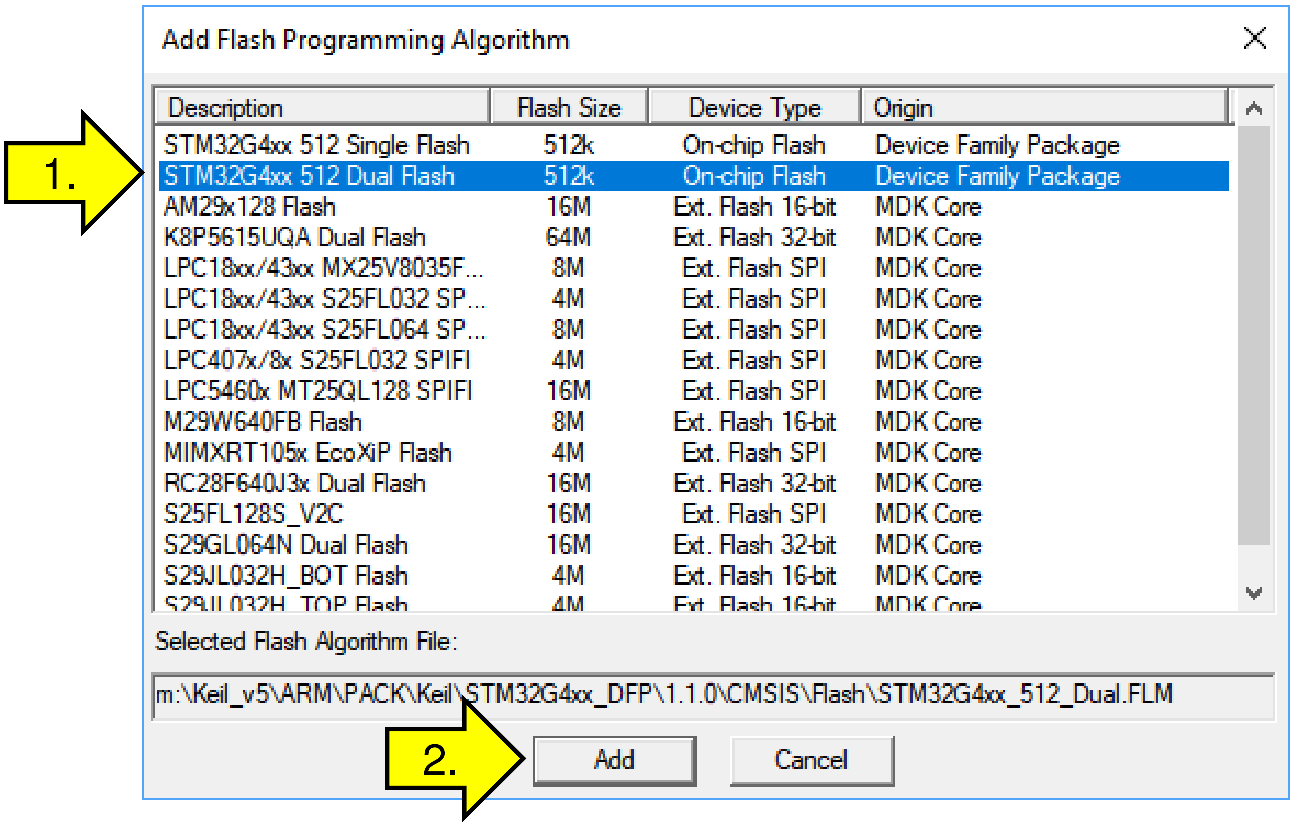
ST-Link Debugger - Settings



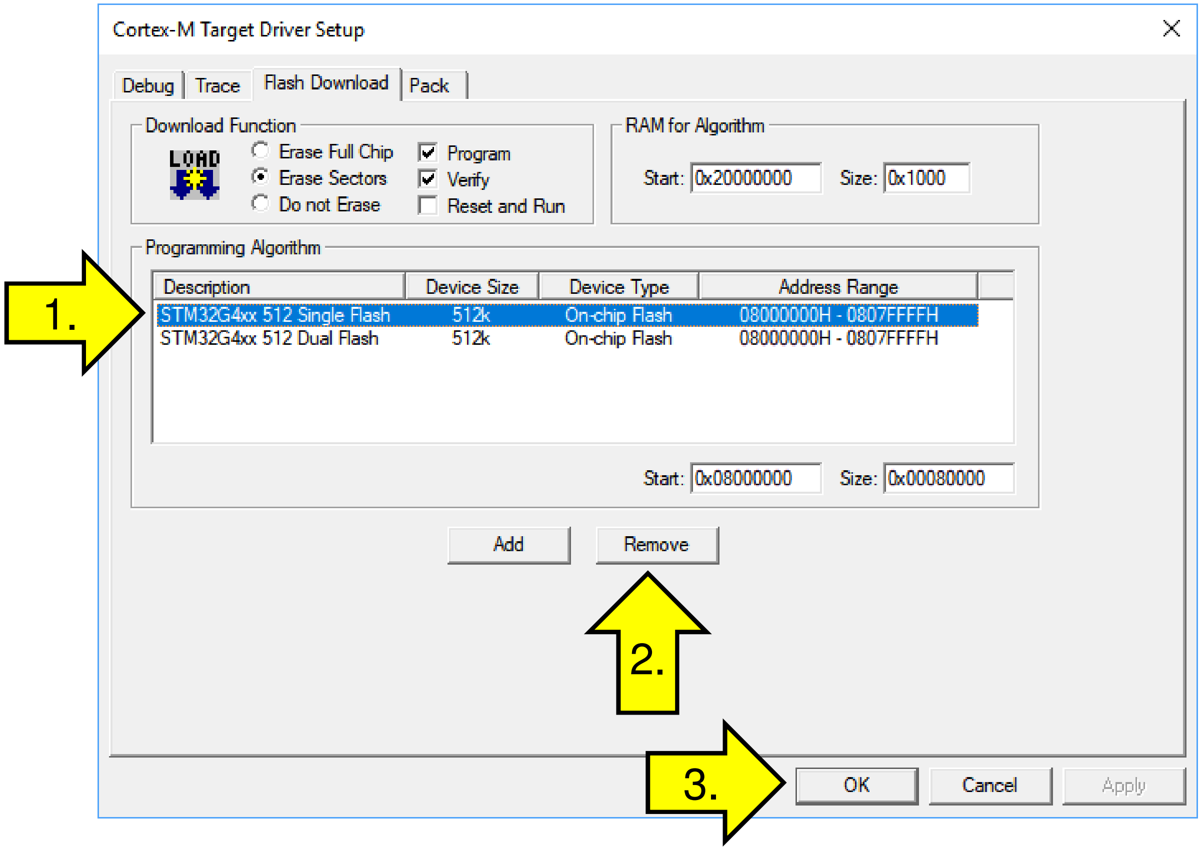
Flash Download → Add



STM32G4xx 512 Dual Flash

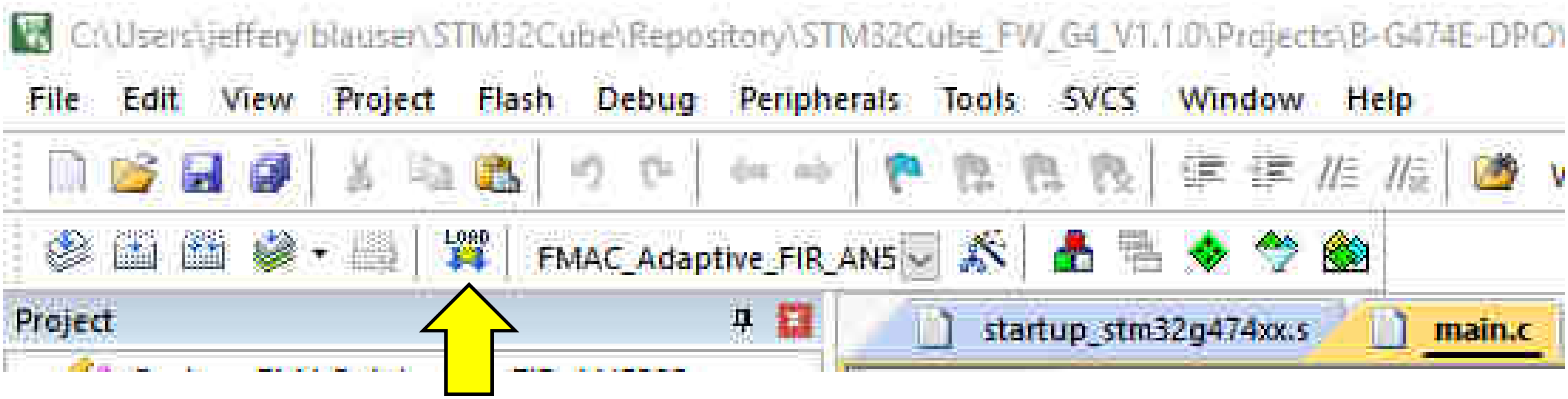


Remove Single Flash



Keil MDK-ARM

Load

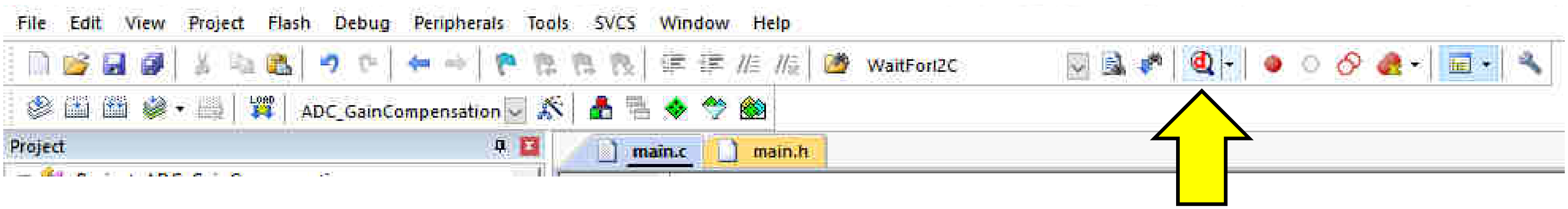


Build Output

```
Build Time Elapsed:  00:01:33
Load "HRTIM_Buck_Boost\\HRTIM_Buck_Boost.axf"
Erase Done.
Programming Done.
Verify OK.
Flash Load finished at 18:38:15
```

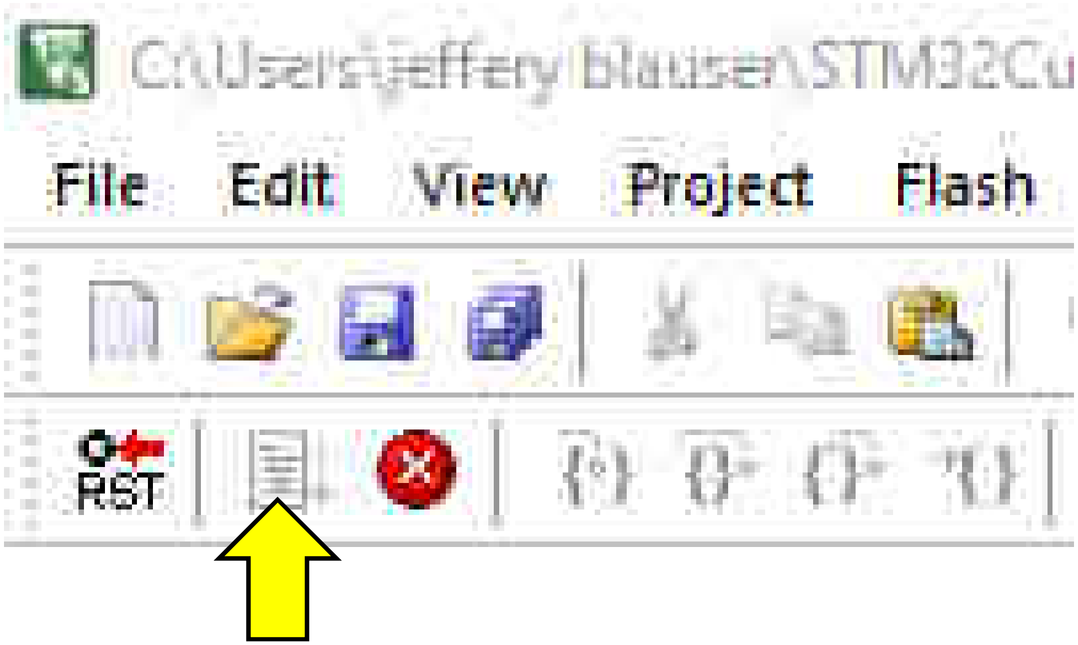
Keil MDK-ARM

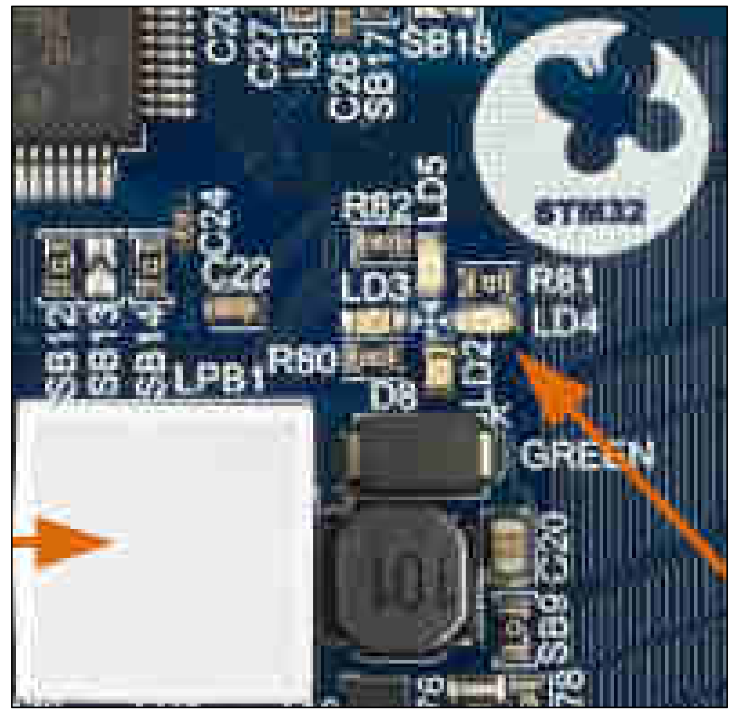
Debugger



Keil MDK-ARM

RUN – Operating as a Buck converter.



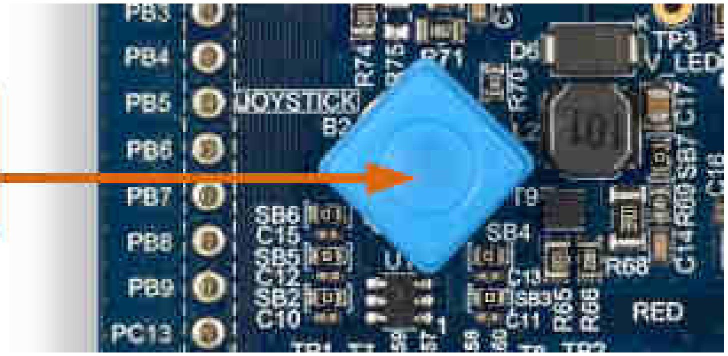


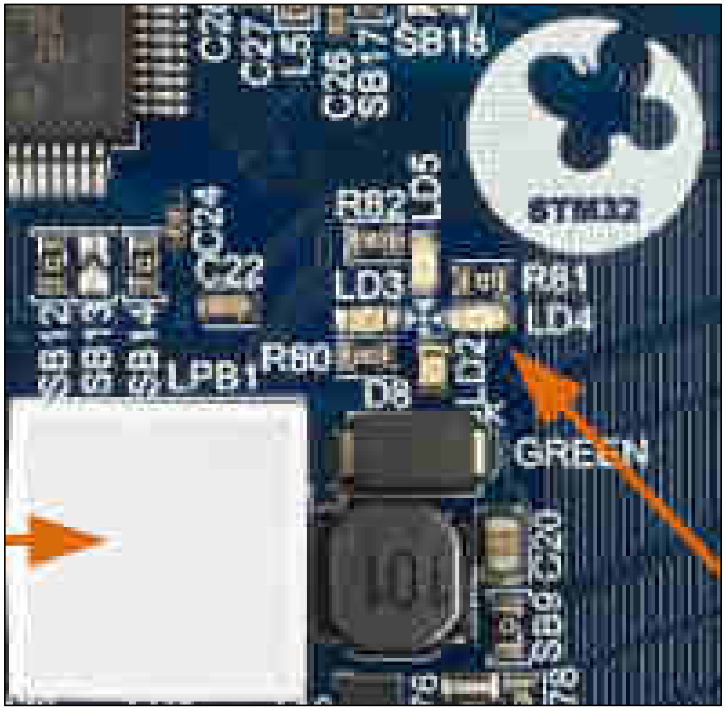
LD4 blinks rapidly

Keil MDK-ARM

Press the Blue User Button for BOOST operation

B2
Joystick

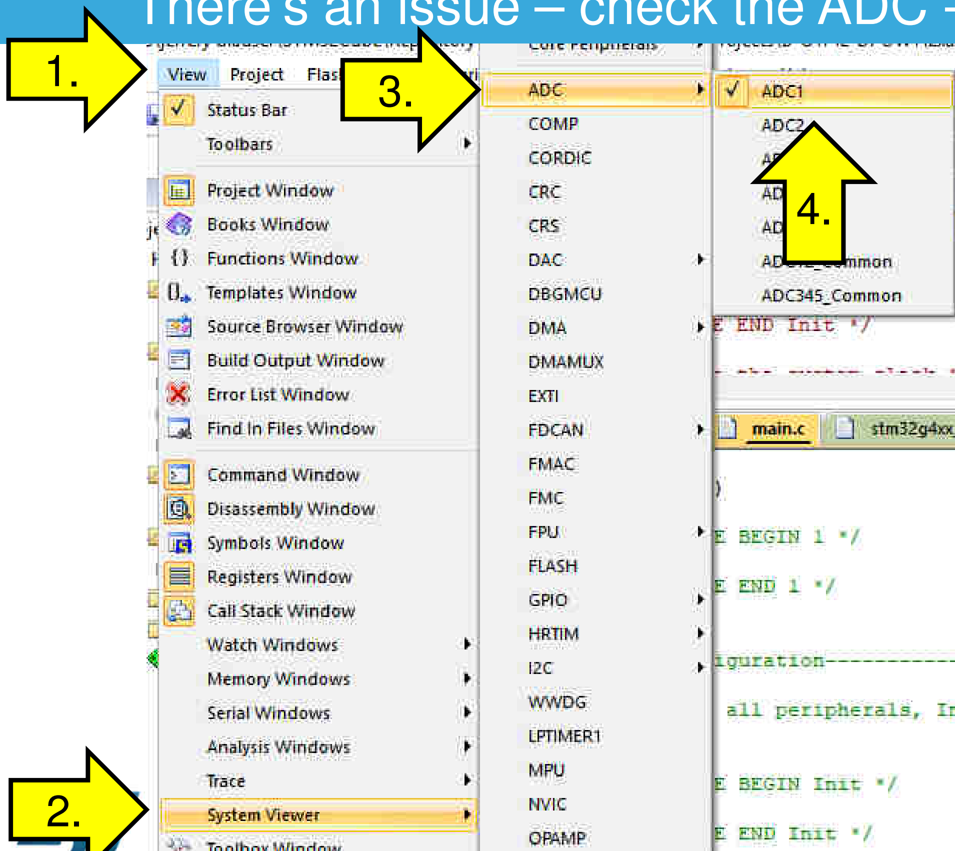




LD3 blinks rapidly

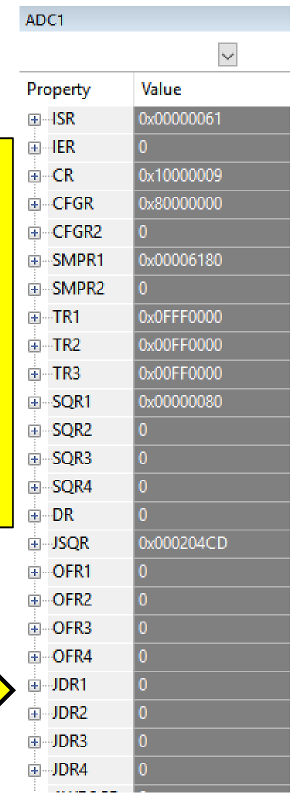
Keil MDK-ARM

There's an issue – check the ADC - VIEW→System Viewer→ADC→ADC1



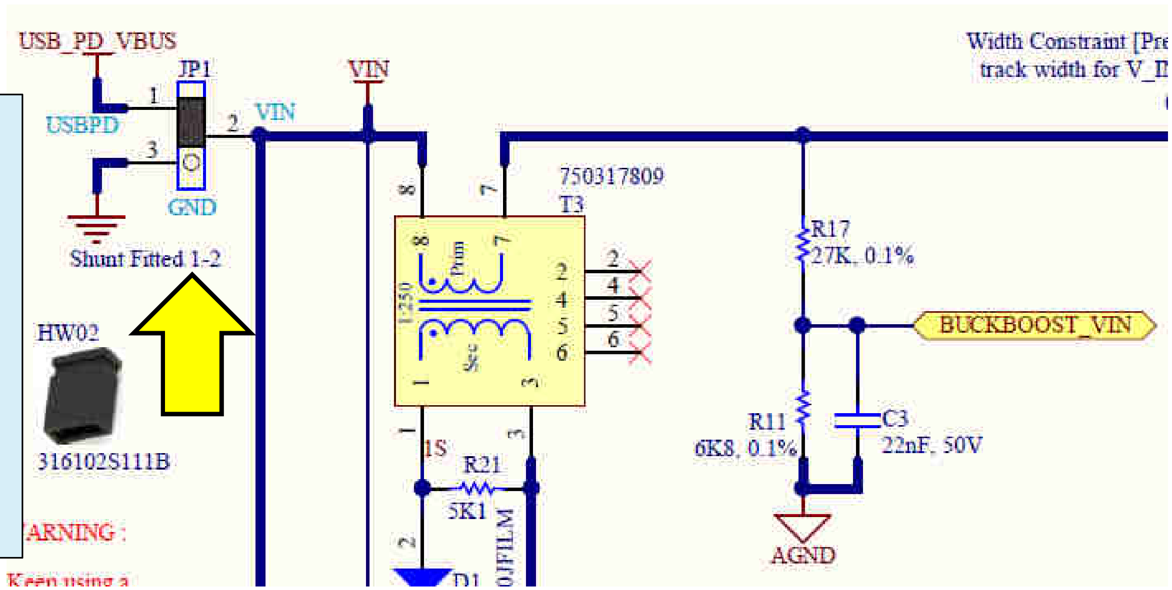
There are zeros in ADC1→JDR1 & 2 which are the two injected channel data registers

Uh Oh



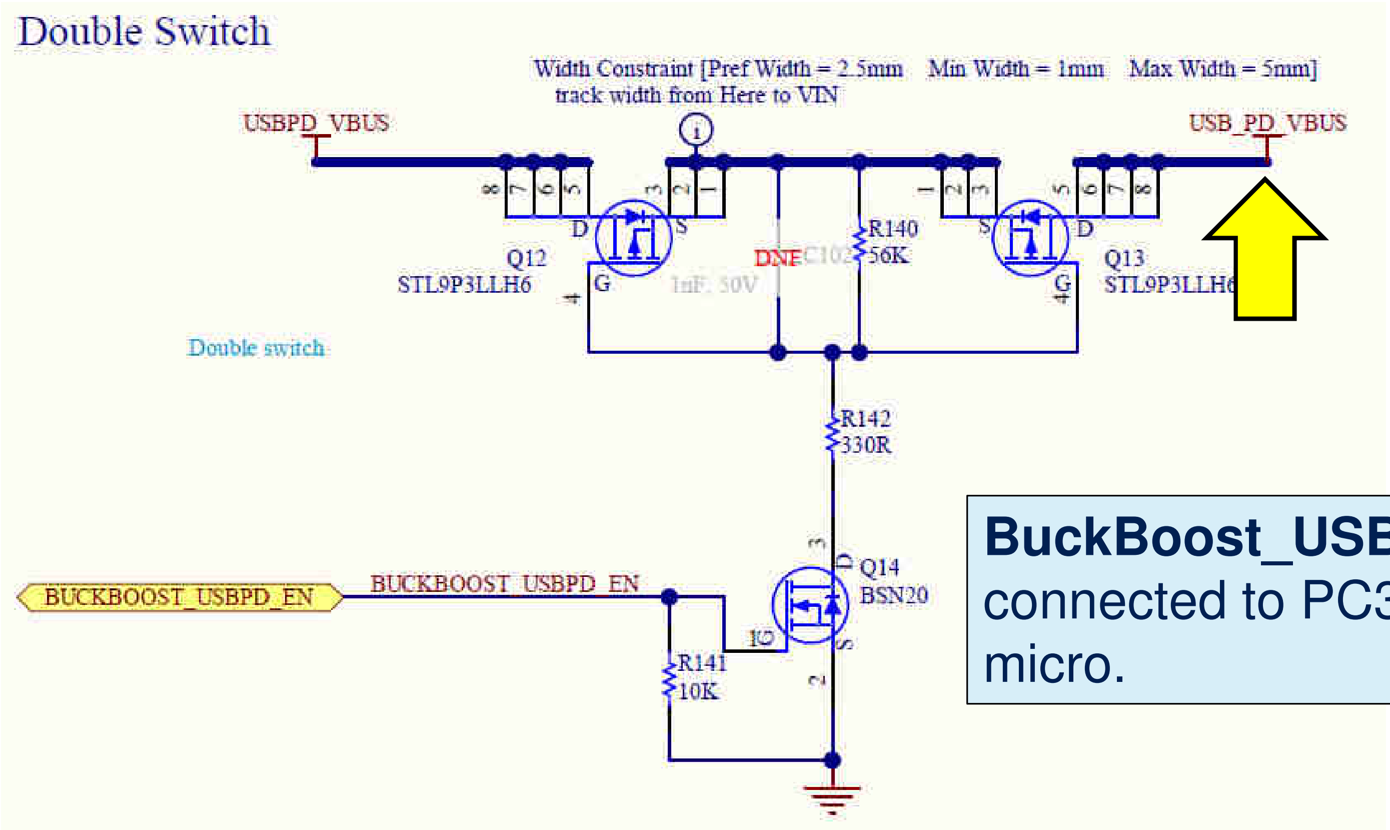
Schematic

5V from USB-C is available here. Are JP1 pins 1 & 2 jumpered together? Keep **USB_PD_VBUS** in mind.



BuckBoost_VIN is connected to ADC1_IN2.

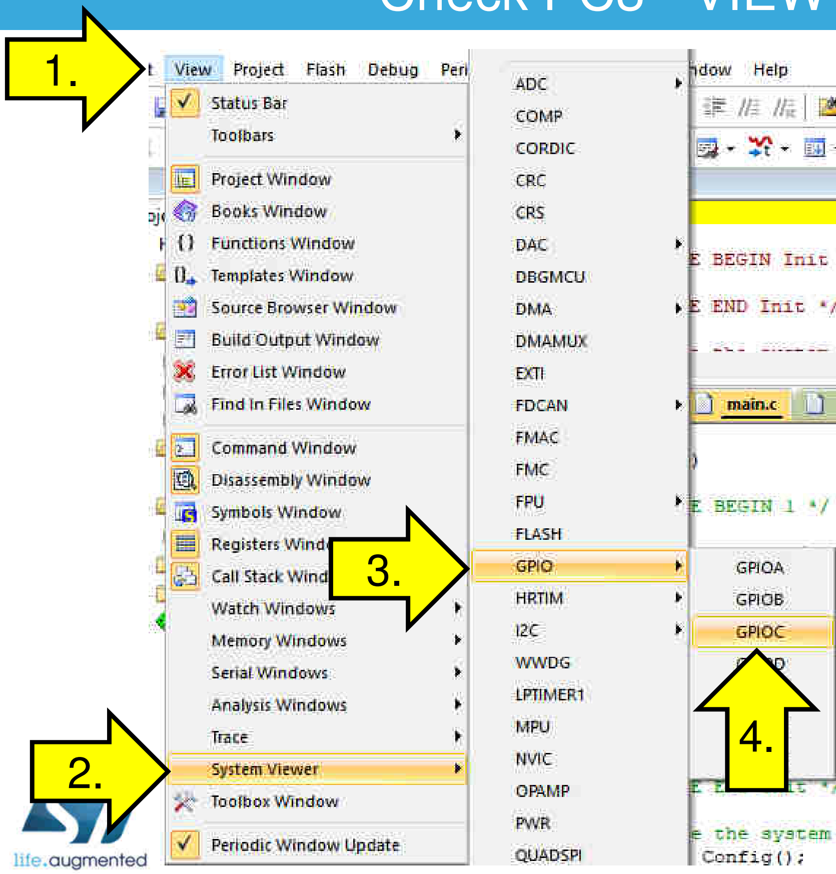
Schematic – Where do we find USB_PD_VBUS?



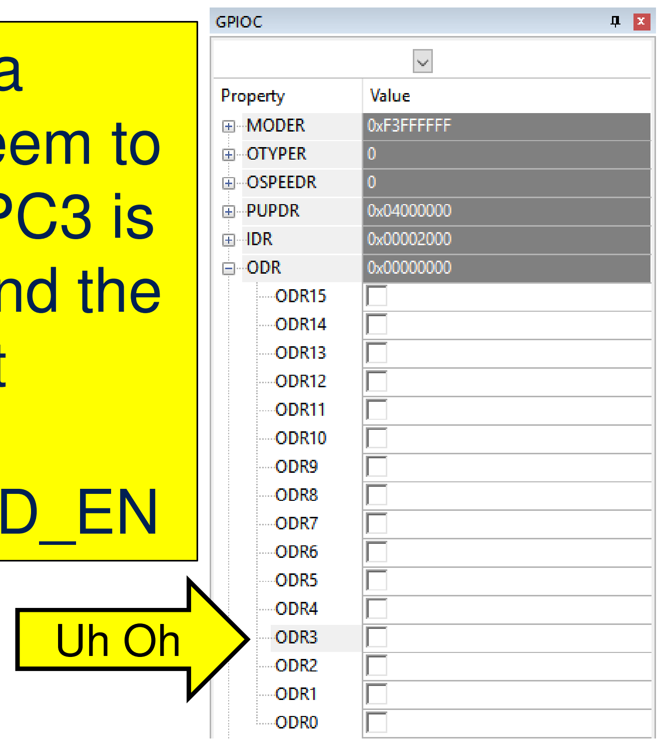
BuckBoost_USBPD_EN is connected to PC3 of the micro.

Keil MDK-ARM

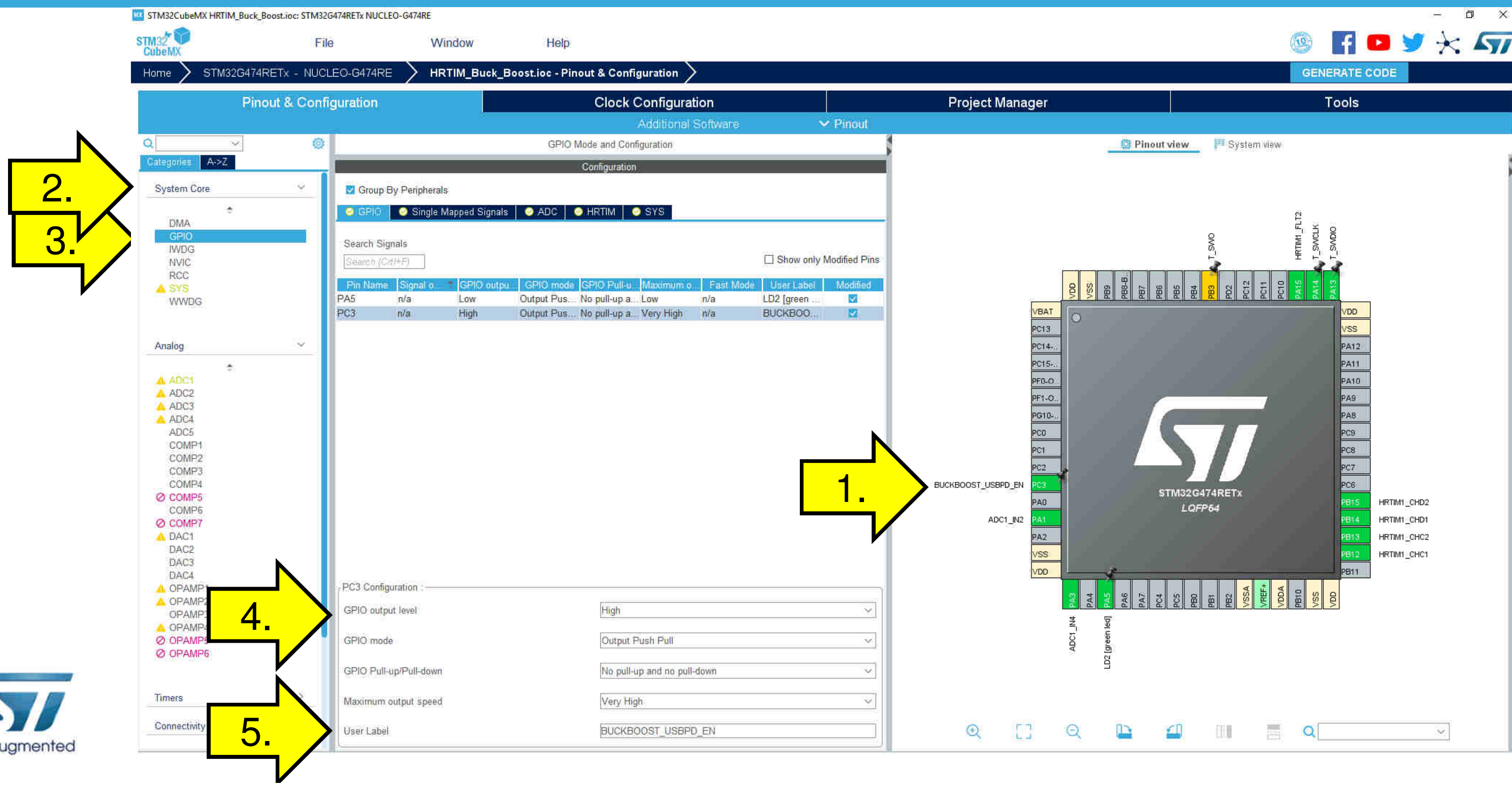
Check PC3 - VIEW→System Viewer→GPIO→GPIOC



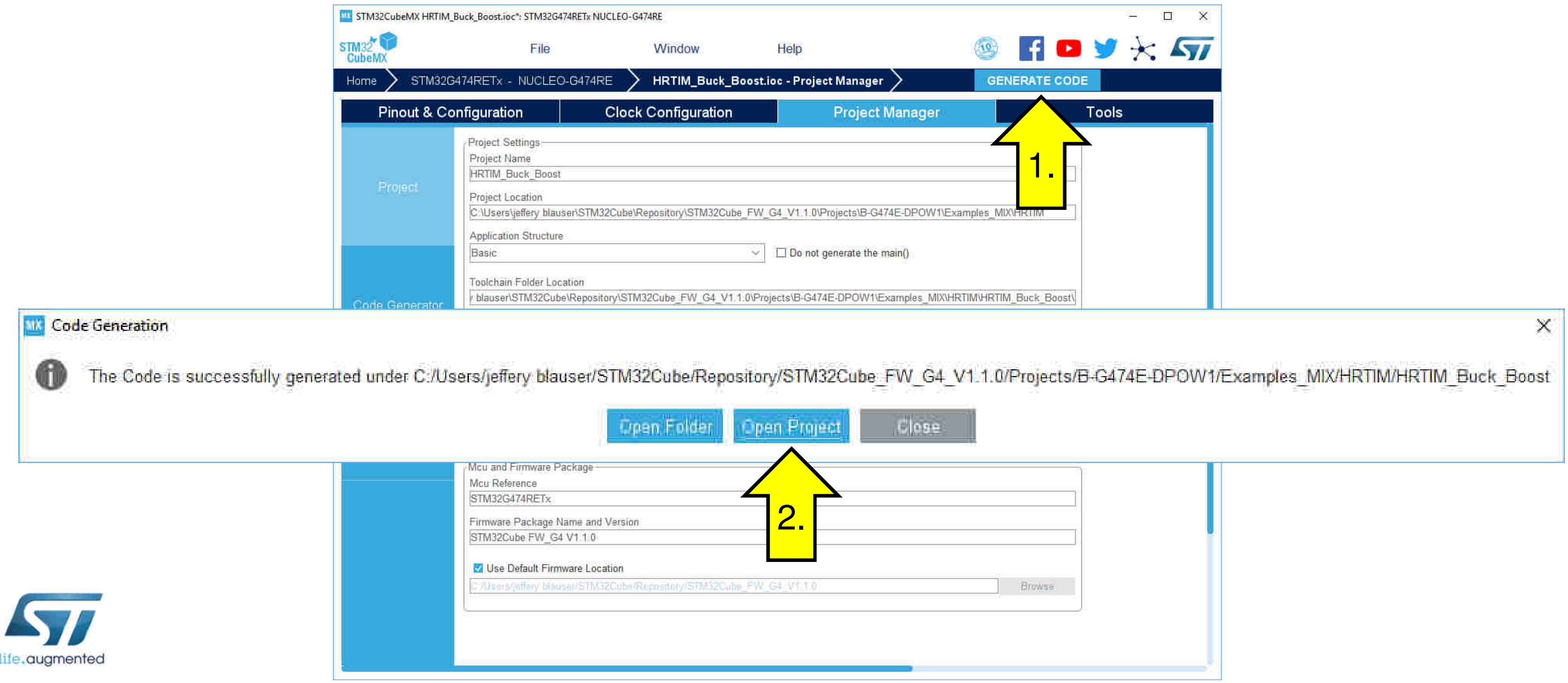
GPIOC output data registers do not seem to be configured so PC3 is not being driven and the MOSFETS are not enabled by BuckBoost_USBPD_EN



Click on PC3→Choose GPIO_OutPut→System Core→GPIO



Generate Code

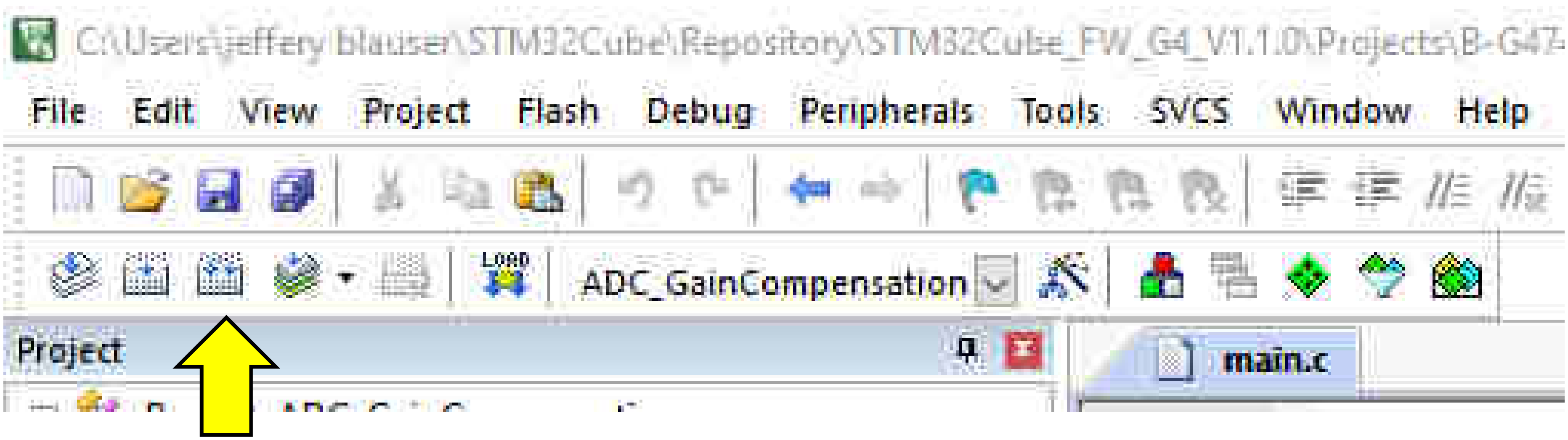


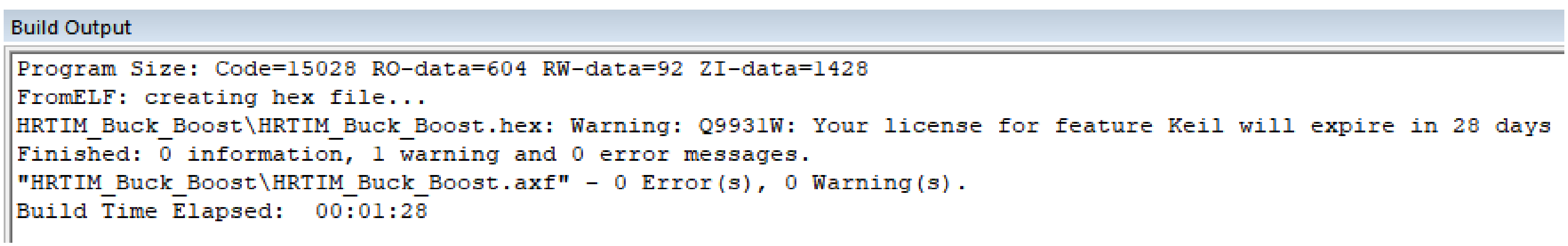
Main.c with new GPIOC code (you do not add this code)

```
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};

  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOC_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(BUCKBOOST_USBPD_EN_GPIO_Port, BUCKBOOST_USBPD_EN_Pin, GPIO_PIN_SET);

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, GPIO_PIN_RESET);

  /*Configure GPIO pin : BUCKBOOST_USBPD_EN_Pin */
  GPIO_InitStruct.Pin = BUCKBOOST_USBPD_EN_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
  HAL_GPIO_Init(BUCKBOOST_USBPD_EN_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pin : LD2_Pin */
  GPIO_InitStruct.Pin = LD2_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(LD2_GPIO_Port, &GPIO_InitStruct);

}
```

Keil MDK-ARM

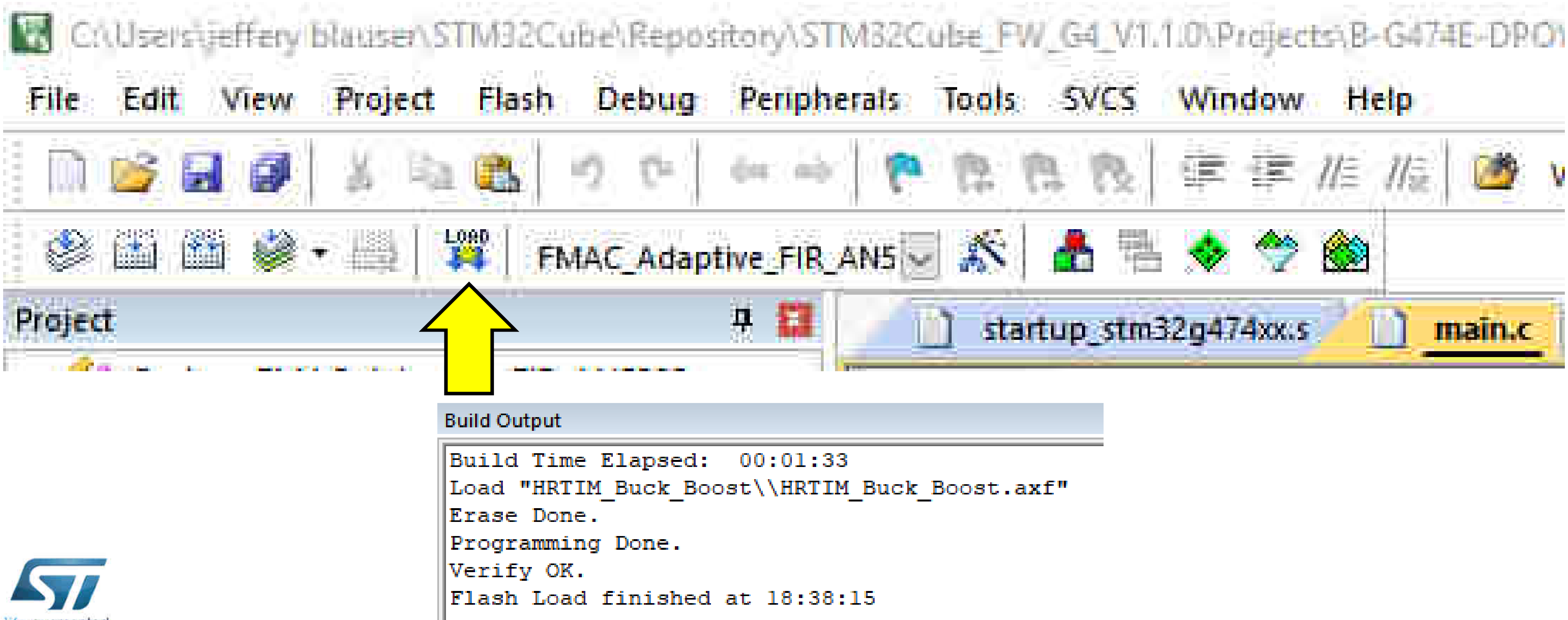
Build



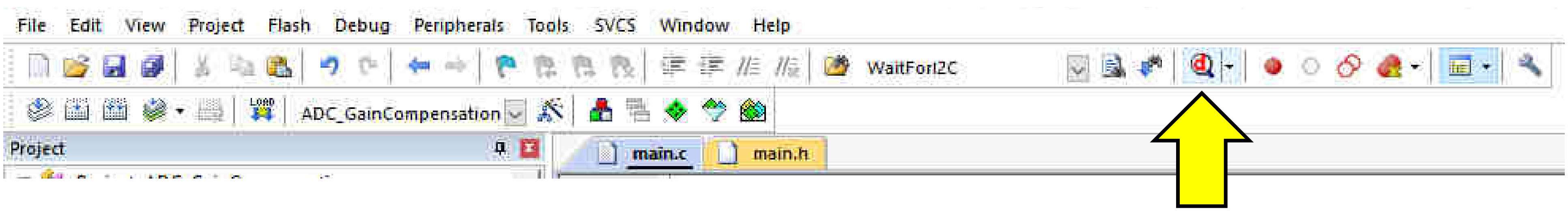


Keil MDK-ARM

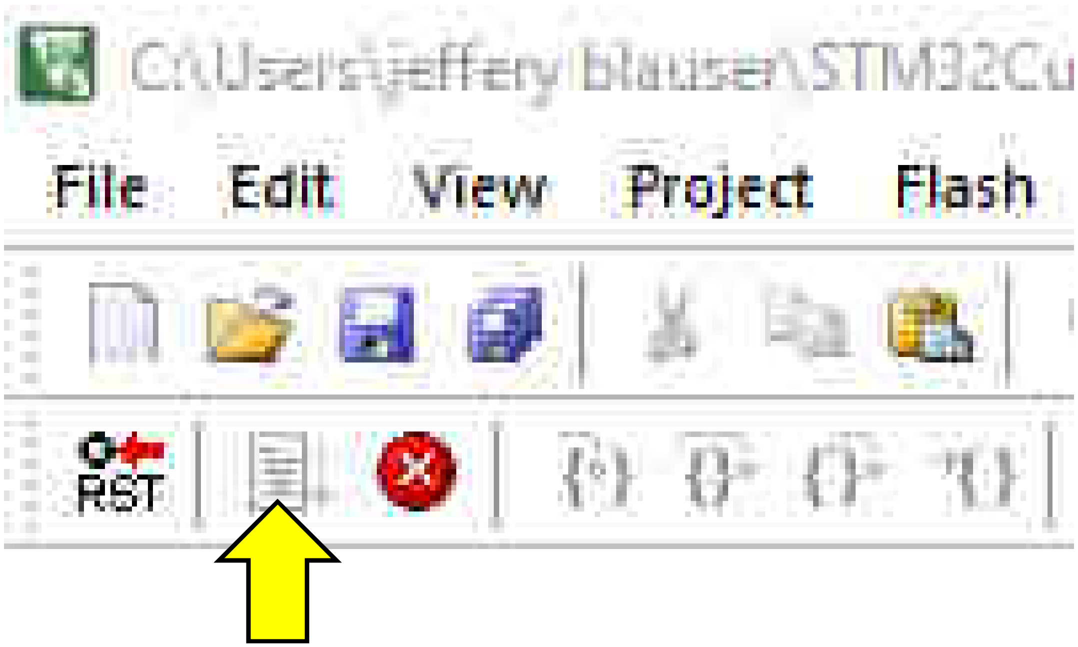
Load

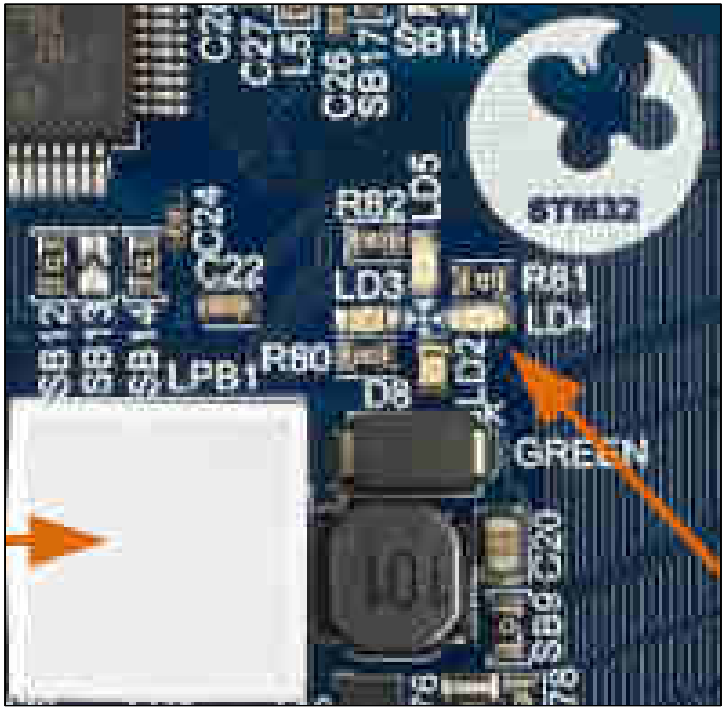


Debugger


File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
WaitForI2C
ADC_GainCompensation
Project
main.c
main.h

RUN – Operating as a Buck converter.

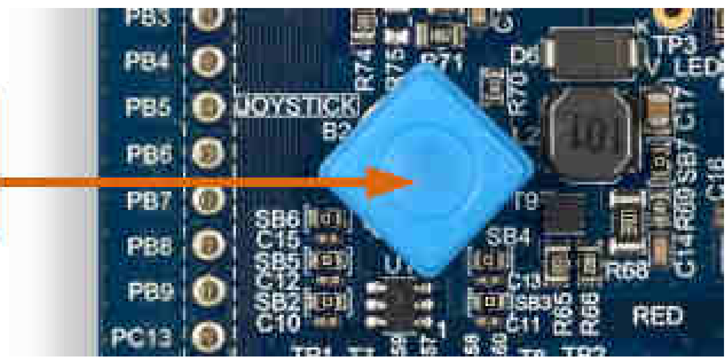


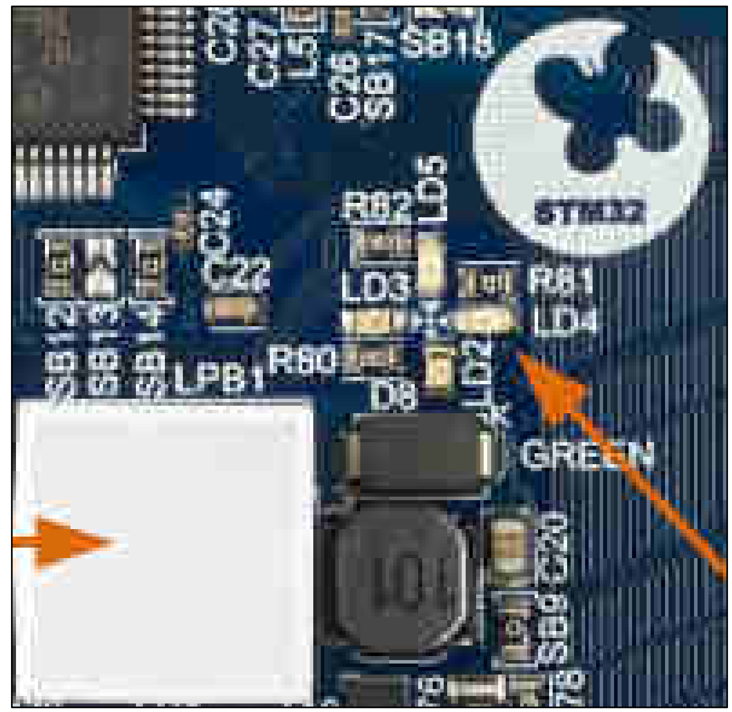


LD4 blinks rapidly

Press the Blue User Button for BOOST operation

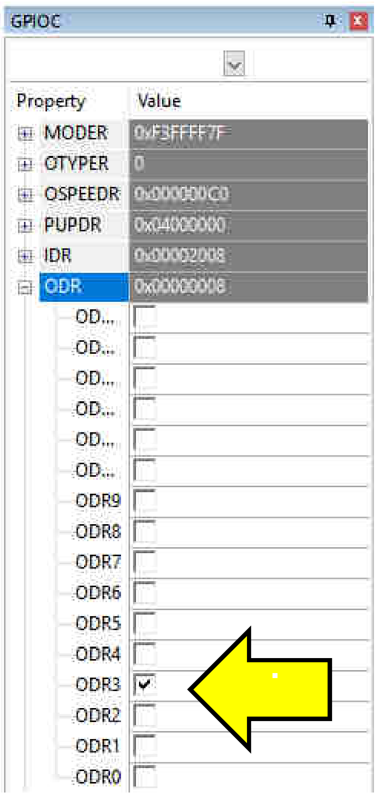
B2
Joystick

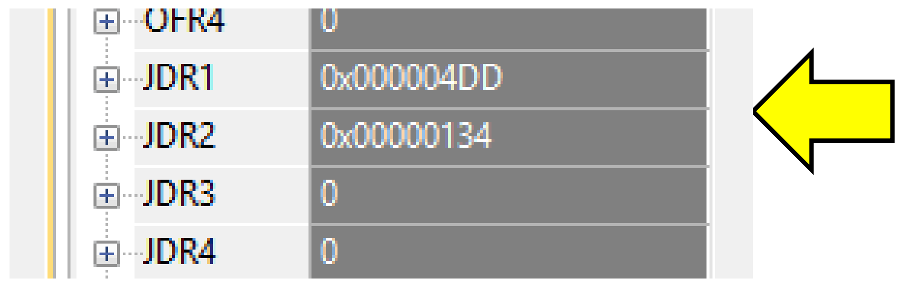




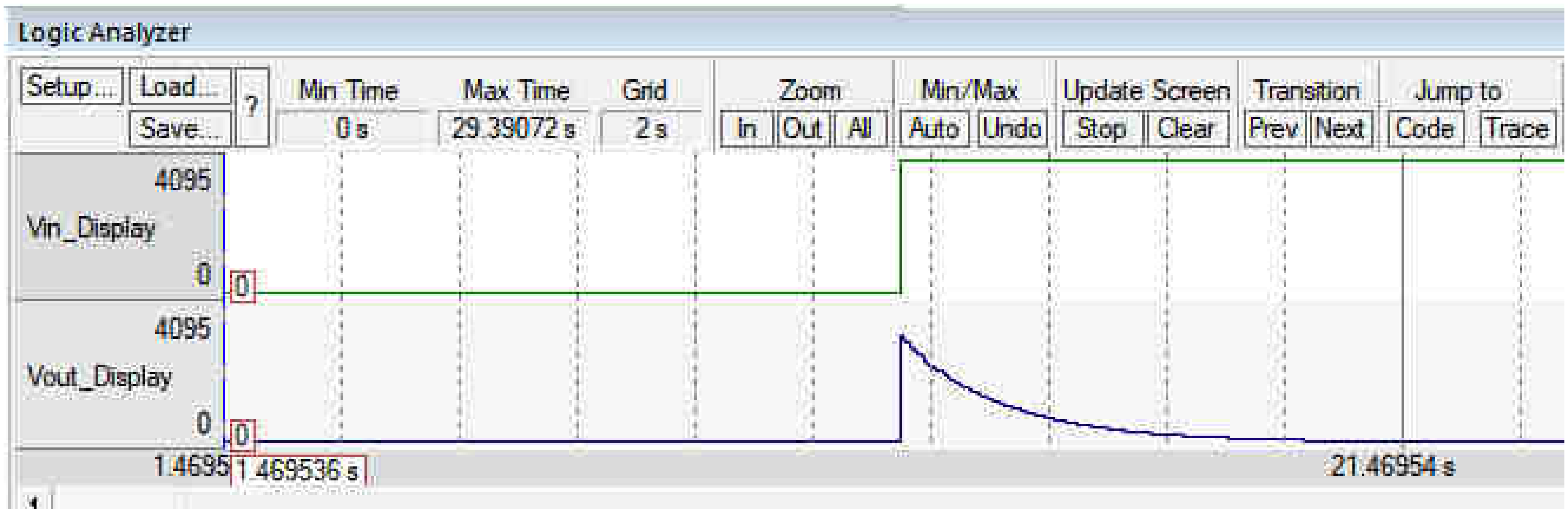
LD3 blinks rapidly

Check GPIOC – PC3 and ADC1 injected data registers

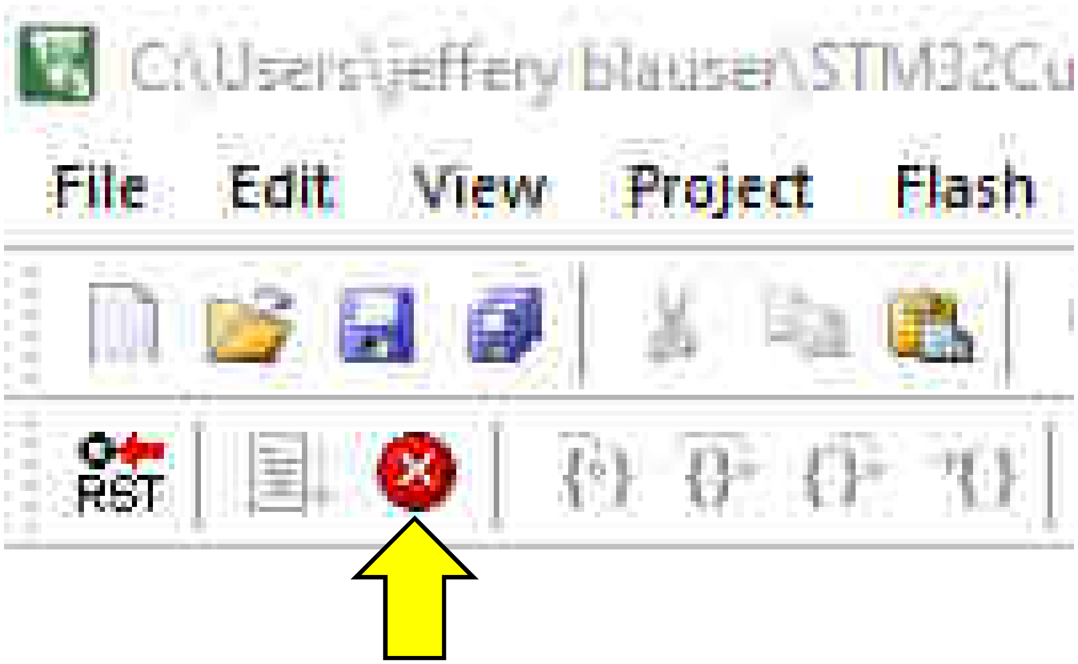




Logic Analyzer – Vin & Vout – Buck Mode

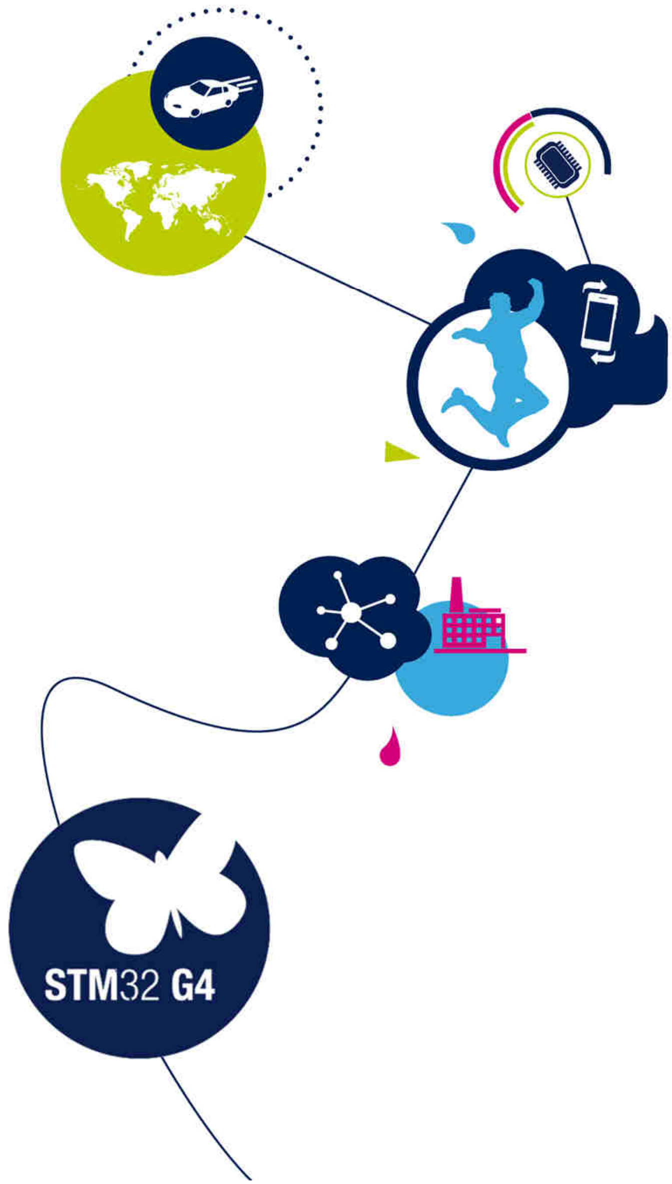


STOP Execution



STM32G4 mixed-signal MCU workshop

Hands On: CORDIC Hardware Accelerator



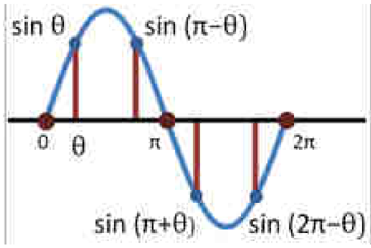
life.augmented

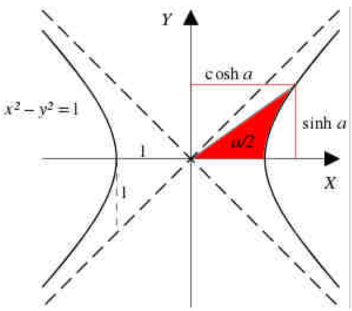
CORDIC: Main features

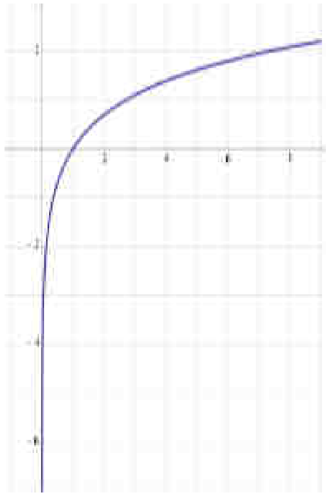
- 24-bit CORDIC rotation engine
  - Trigonometric and Hyperbolic functions
- Circular and Hyperbolic modes
- Rotation and Vectoring modes
- Programmable precision
- Low latency AHB slave interface
- DMA read and write channels

Cordic: Main features

- Supported functions:
  - Sine, Cosine: Vector rotation (polar to rectangular)
  - Arctanangent, Modulus: Vector translation (rectangular to polar)
  - Hyperbolic cosine
  - Hyperbolic sine
  - Hyperbolic tangent
  - Natural logarithm
  - Square root







- Fixed point signed input and output
  - Q1.31 or Q1.15 format
- Performance improvement relative to software
  - Sine and Cosine functions are typically 5x faster than software execution

life.augmented

CORDIC - Precision

- Precision is dependent on number of iterations

Function	Number of iteration (precision)	Number of cycles	Max residual error	
			q1.31 format	q1.15. format
Sin, Cos, Phase, Mod, Atan	4	1	2^{-3}	2^{-3}
	8	2	2^{-7}	2^{-7}
	12	3	2^{-11}	2^{-11}
	16	4	2^{-15}	2^{-15}
	20	5	2^{-18}	2^{-16}
	24	6	2^{-19}	2^{-16}

life.augmented

CORDIC - Precision

Function	Number of iteration (precision)	Number of cycles	Max residual error	
			q1.31 format	q1.15. format
Sinh, Cosh, Atanh, Ln	4	1	2^{-2}	2^{-2}
	8	2	2^{-6}	2^{-6}
	12	3	2^{-10}	2^{-10}
	16	4	2^{-13}	2^{-13}
	20	5	2^{-17}	2^{-15}
	24	6	2^{-18}	2^{-15}
Sqrt	4	1	2^{-7}	2^{-7}
	8	2	2^{-14}	2^{-14}
	12	3	2^{-19}	2^{-15}

life.augmented

CORDIC accelerator

Objective:

- To show how to configure the CORDIC accelerator.
  - Calculate the SIN and COS of angles with 6 cycles of precision.
- Enable the Event Recorder to measure the time for the CORDIC and ARM Math Library functions.
- Display CORDIC results in the Logic Analyzer.

CORDIC configuration

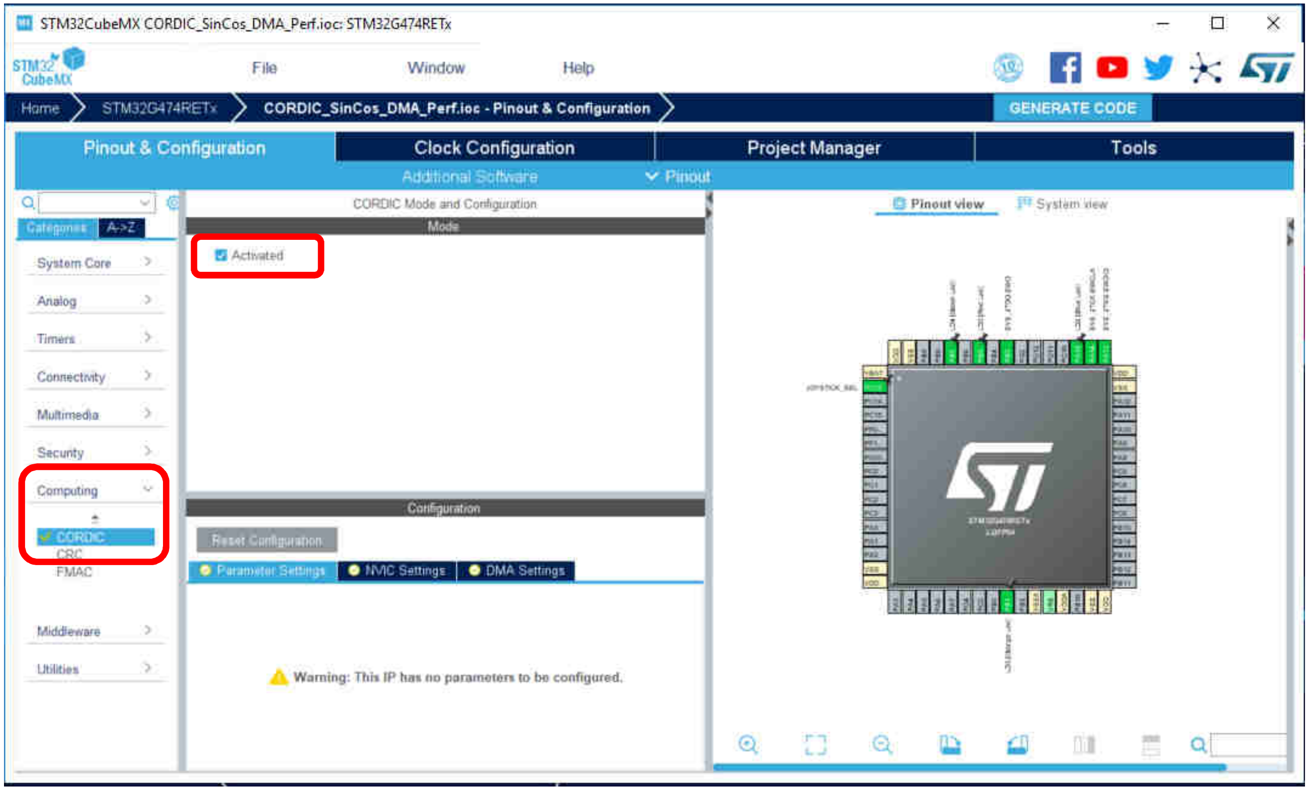
- Open CubeMX project “CORDIC_SinCos_DMA_Perf.ioc” file
  - \labs\Cordic\CORDIC_SinCos_DMA_Perf

Name	Date modified	Type	Size
Drivers	8/28/2019 4:57 PM	File folder	
Inc	8/28/2019 8:22 PM	File folder	
MDK-ARM	8/29/2019 2:53 AM	File folder	
Src	8/28/2019 8:22 PM	File folder	
.mxproject	8/28/2019 8:22 PM	MXPROJECT File	7 KB
CORDIC_SinCos_DMA_Perf.ioc	8/29/2019 3:32 AM	IOC File	9 KB

life.augmented

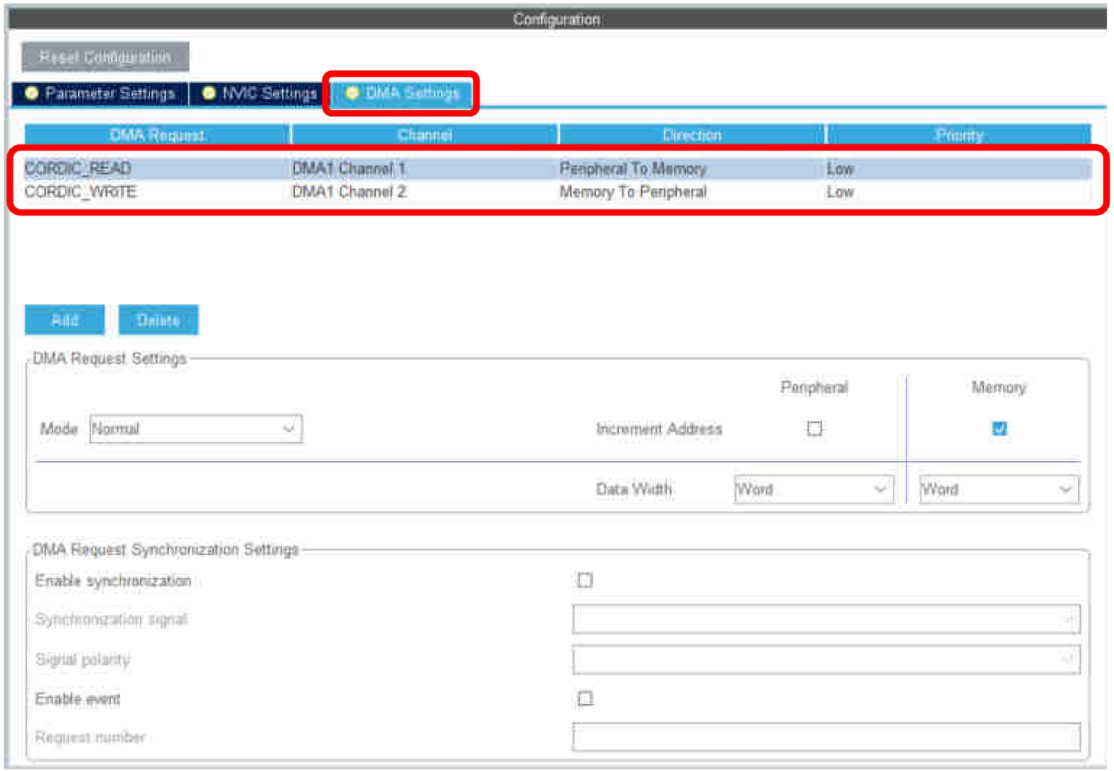
CORDIC configuration

- Expand “Computing” Category
- Select “CORDIC”
  - CHECK “Activated”



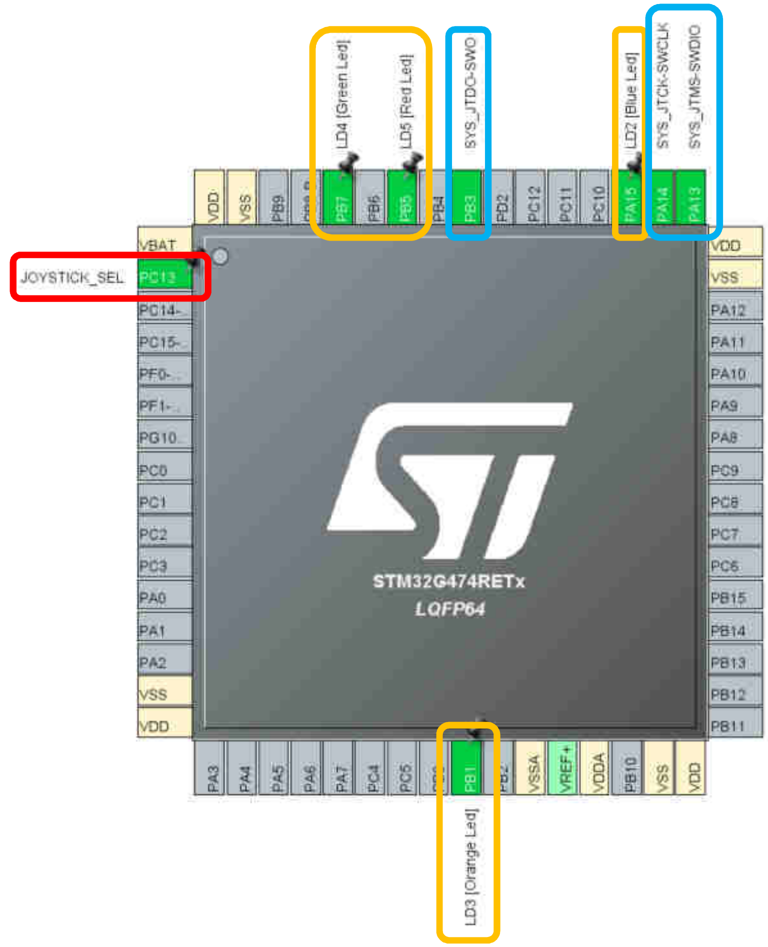
CORDIC configuration

- DMA Request is enabled to transfer data in and out of CORDIC
- 32-bit Data Width (Word)



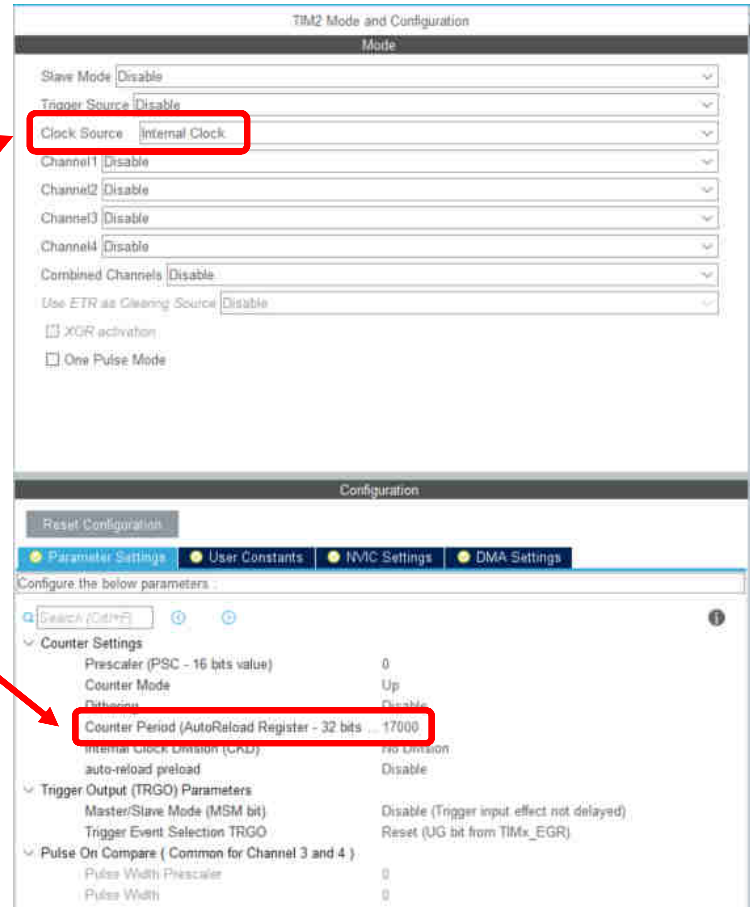
Additional Peripheral configuration

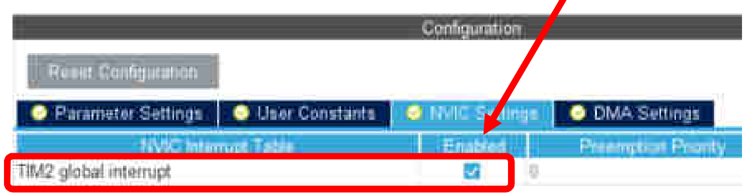
- JOYSTICK_SEL: EXTI13
  - PC13
- LD2-5: GPIO_Output
  - PA15, PB1, PB5 and PB7
- Serial Wire Debug with Trace (SWO):
  - PA13, PA14 and PB3



Additional Peripheral configuration

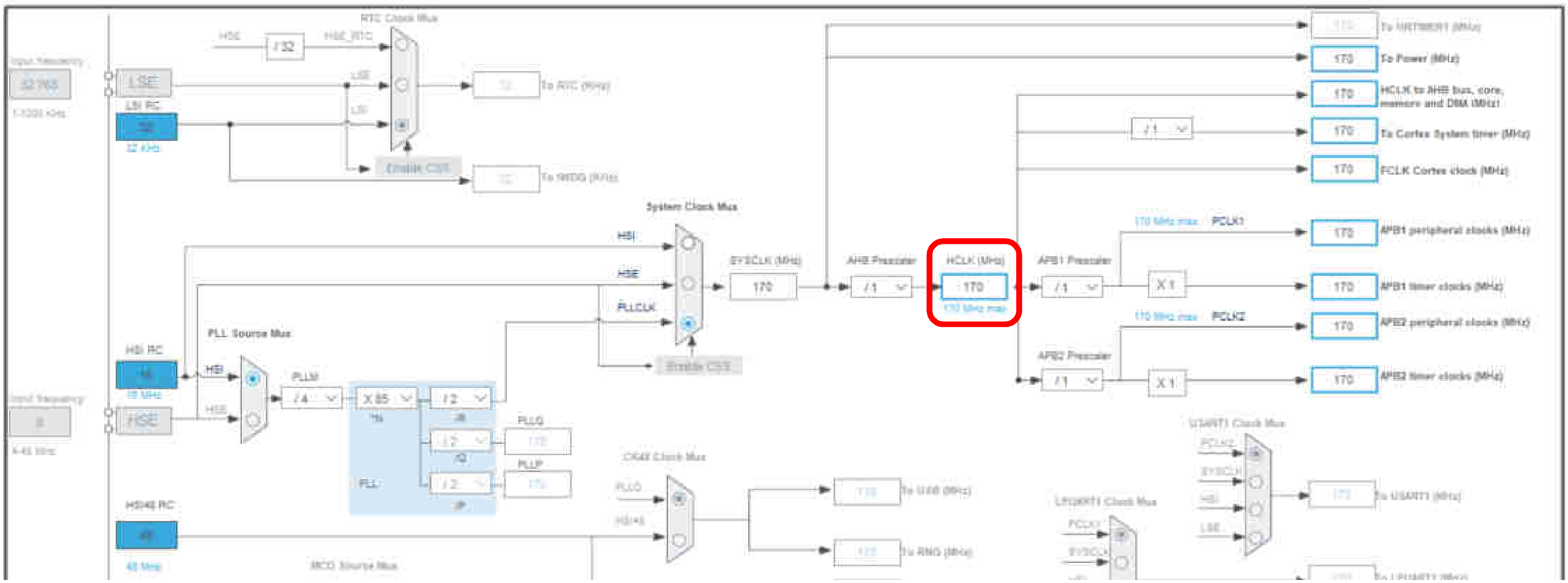
- Timer 2: Debug Logic Analyzer
- Internal Clock Source = 170 MHz
- Counter Period = 17000
  - 10 kHz timebase interrupt
- TIM2 global interrupt Enabled





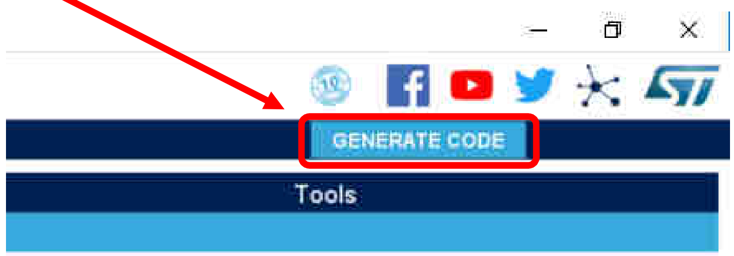
Hands-On: CORDIC

- Validate “HCLK (MHz)” is set to 170



Hands-On: CORDIC

- Generate Code



- Open Project in MDK-ARM

Hands-On: CORDIC

- aAnglesCordic - Array of angles for CORDIC Q1.31 format (Line 57)

```
56   /* Array of angles for CORDIC Q1.31 format, regularly incremented from 0 to 2*pi */
57   static int32_t aAnglesCordic[ARRAY_SIZE] =
58   {
59     0x00000000, 0x04000000, 0x08000000, 0x0C000000,
```

- aAnglesLib – Array of angles for ARM Math library (Line 78)

```
77   /* Array of angles for arm math library Q1.31 format, regularly incremented from 0 to 2*pi */
78   static int32_t aAnglesLib[ARRAY_SIZE] =
79   {
80     0x00000000, 0x02000000, 0x04000000, 0x06000000,
```

- aRefSin & aRefCos – Array of Sin & Cos result reference (Line 99 & 120)

```
98   /* Array of reference sines in Q1.31 format */
99   static int32_t aRefSin[ARRAY_SIZE] =
100  {
101    0x00000000, 0x0C8BD35E, 0x18F8B83C, 0x25280C5D,
```

```
119  /* Array of reference cosines in Q1.31 format */
120  static int32_t aRefCos[ARRAY_SIZE] =
121  {
122    0x80000000, 0x7F62368F, 0x7D8A5F3F, 0x7A7D055B,
```

- aResults – Array of calculation results (Line 142)

```
140  /* Array of calculation results in Q1.31 format.
141     Will contain alternatively Sine and Cosine of input angles */
142  static int32_t aResults[2 * ARRAY_SIZE];
```

CORDIC configuration

- Sine calculation function (Line 219)
- 6 cycles of precision (Line 220)
- No Scaling (Line 221)
- One input angle data (Line 222)
  - Q1.31 format for input data (Line 224)
- Two output data (Sine and Cosine) (Line 223)
  - Q1.31 format for input data (Line 225)

```
217  /* USER CODE BEGIN 2 */
218  /*##-2- Configure the CORDIC peripheral ##################
219  sCordicConfig.Function         = CORDIC_FUNCTION_SINE;
220  sCordicConfig.Precision        = CORDIC_PRECISION_6CYCLES;
221  sCordicConfig.Scale            = CORDIC_SCALE_0;
222  sCordicConfig.NbWrite          = CORDIC_NBWRITE_1;
223  sCordicConfig.NbRead           = CORDIC_NBREAD_2;
224  sCordicConfig.InSize           = CORDIC_INSIZE_32BITS;
225  sCordicConfig.OutSize          = CORDIC_OUTSIZE_32BITS;
```

CORDIC configuration

- Step = 0
  - Calculation made with CORDIC
  - Simple call to HAL_CORDIC_Calculate_DMA

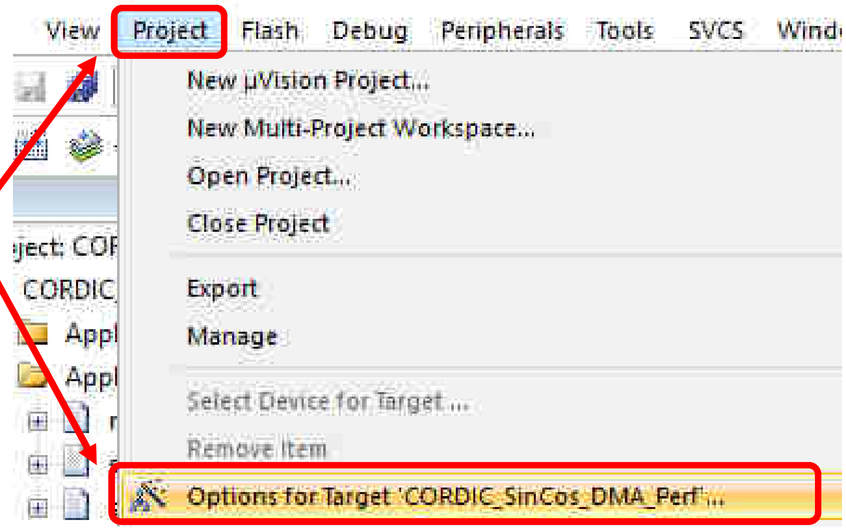
```c
/*################### Calculation using CORDIC #######################*/
case 0:
  for (uint32_t i = 0; i < LOOP_NB; i++)
  {

    /* Start calculation of sines in DMA mode */
    if (HAL_CORDIC_Calculate_DMA(&hcordic, aAnglesCordic, aResults,
                                 ARRAY_SIZE, CORDIC_DMA_DIR_IN_OUT) != HAL_OK)
    {
      /* Processing Error */
      Error_Handler();
    }

```

- Step = 1
  - Calculation made with ARM Math library
  - Lines 277 through 283 are called to perform same CORDIC operation

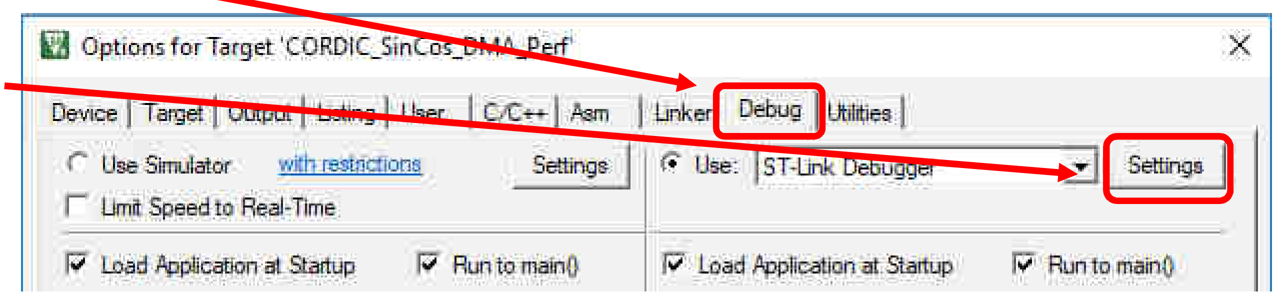
```c
272  /*################### Calculation using arm math library #############*/
273  case 1:
274    for (uint32_t i = 0; i < LOOP_NB; i++)
275    {
276
277      for (uint32_t j = 0; j < ARRAY_SIZE; j++)
278      {
279        /* Calculate sine */
280        aResults[2*j] = arm_sin_q31(aAnglesLib[j]);
281
282        /* Calculate cosine */
283        aResults[(2*j) + 1] = arm_cos_q31(aAnglesLib[j]);
284      }
285
```

Enable Debug Trace

- Open “Options for Target ‘CORDIC_SinCos_DMA_Perf’ …” window.

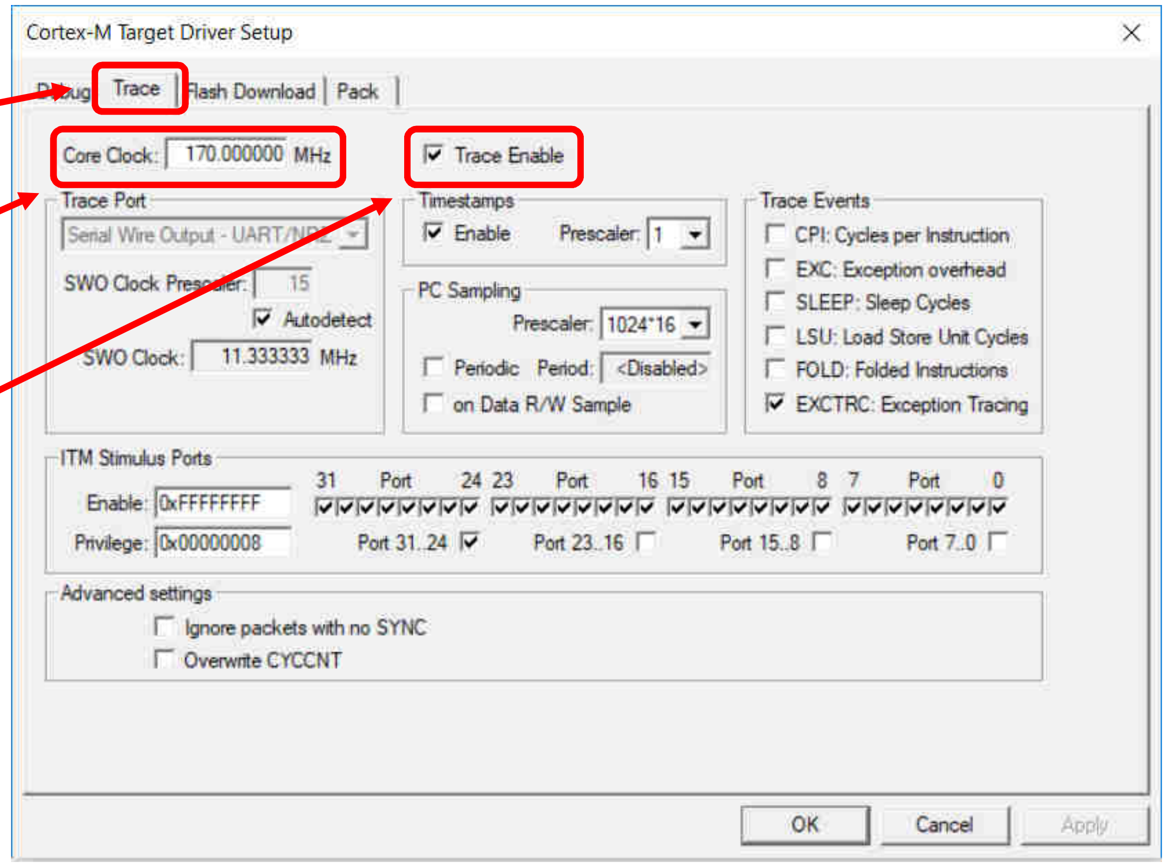


- Switch to “Debug” tab
- Click “Settings” button



Enable Debug Trace

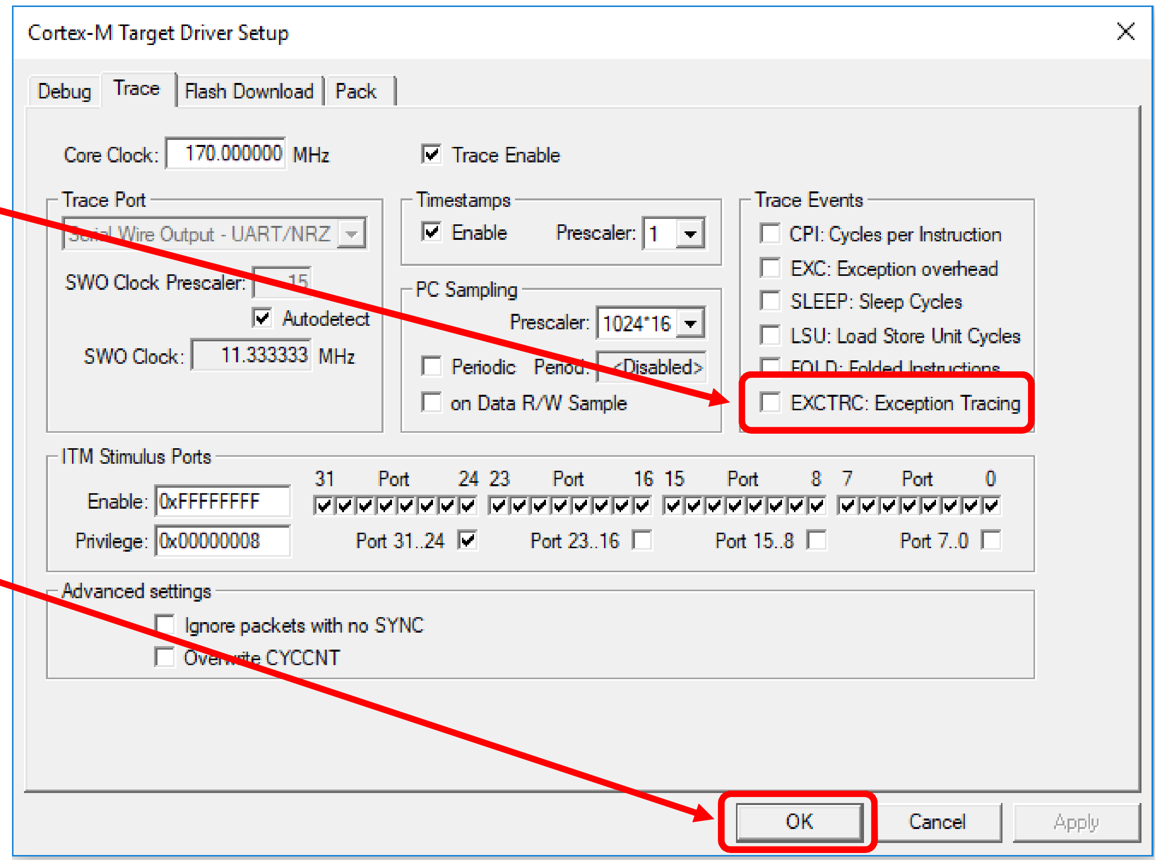
- Switch to “Trace” tab
- Set “Core Clock” = 170 MHz
- Click “Trace Enable” check box



Enable Debug Trace

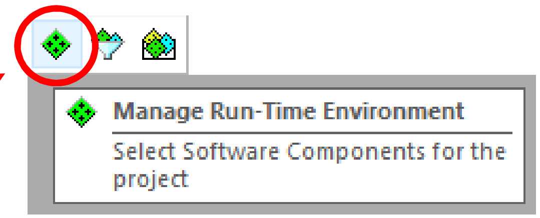
- Disable “Exeception Tracing”
  - Uncheck Box

- Click “OK” to close setup window

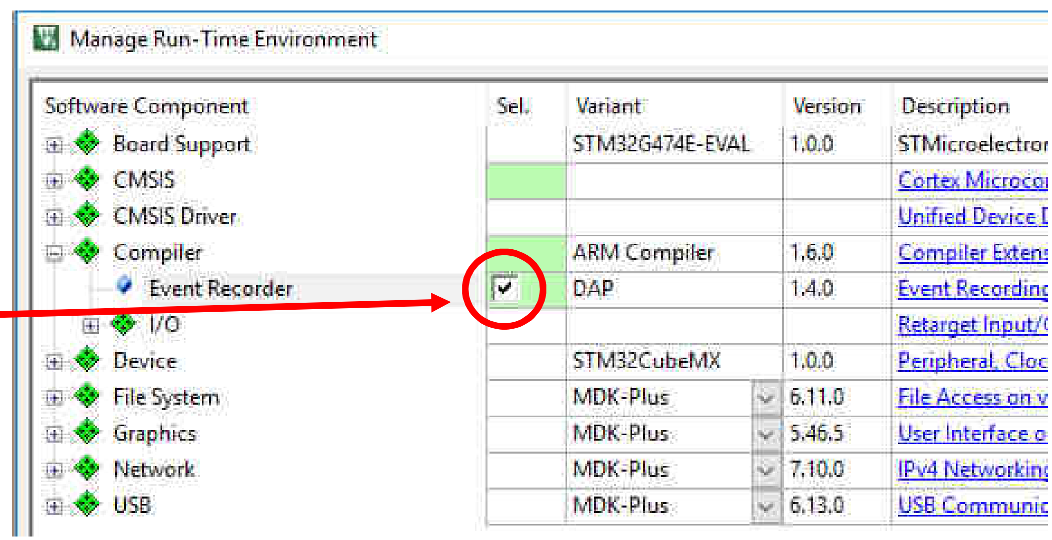


Enable Event Recorder

- Click on Manage Run-Time Environment

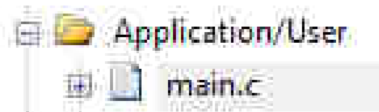


- Expand “Compiler”
- Enable “Event Recorder”
  - Check Box
- Click “OK”



Enable Event Recorder

- Open “main.c” under Application/User folder



- Line 27: Add (uncomment)

#include “EventRecorder.h

```
24 /* Private includes ----------------------
25 /* USER CODE BEGIN Includes */
26 #include <arm_math.h>
27 #include "EventRecorder.h"
28 /* USER CODE END Includes */
```

- Line 191 & 192: Add (uncomment)

EventRecorderInitialize(EventRecordAll, 1);
EventRecorderStart();

```
178 int main(void)
179 {
180   /* USER CODE BEGIN 1 */
181   /* STM32G4xx HAL library initialization:
182        - Configure the Flash prefetch
183        - Systick timer is configured by default as source of time base, but user
184          can eventually implement his proper time base source (a general purpose
185          timer for example or other time source), keeping in mind that Time base
186          duration should be kept 1ms since PPP_TIMEOUT_VALUEs are defined and
187          handled in milliseconds basis.
188        - Set NVIC Group Priority to 4
189        - Low Level Initialization
190      */
191   EventRecorderInitialize(EventRecordAll, 1);
192   EventRecorderStart();
193   /* USER CODE END 1 */
```

Enable Event Recorder

- Add (uncomment)

Line 250: EventRecord2(0,i,1);
Line 258: EventRecord2(0,i,2);

```
246         /*################### Calculation using CORDIC #######################*/
247         case 0:
248           for (uint32_t i = 0; i < LOOP_NB; i++)
249           {
250             EventRecord2(0,i,1);
251             /* Start calculation of sines in DMA mode */
252             if (HAL_CORDIC_Calculate_DMA(&hcordic, aAnglesCordic, aResults,
253                                          ARRAY_SIZE, CORDIC_DMA_DIR_IN_OUT) != HAL_OK)
254             {
255               /* Processing Error */
256               Error_Handler();
257             }
258             EventRecord2(0,i,2);
259
```

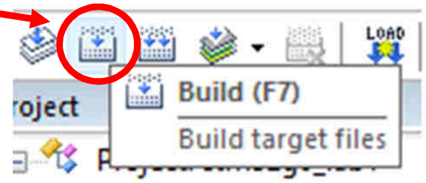
- Add (uncomment)

Line 276: EventRecord2(1,i,1);
Line 285: EventRecord2(1,i,2);

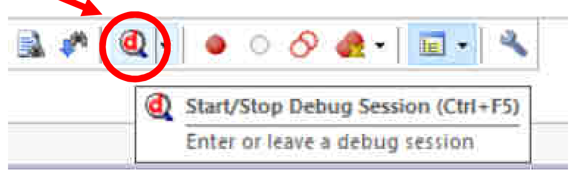
```
272         /*################### Calculation using arm math library #############*/
273         case 1:
274           for (uint32_t i = 0; i < LOOP_NB; i++)
275           {
276             EventRecord2(1,i,1);
277             for (uint32_t j = 0; j < ARRAY_SIZE; j++)
278             {
279               /* Calculate sine */
280               aResults[2*j] = arm_sin_q31(aAnglesLib[j]);
281
282               /* Calculate cosine */
283               aResults[(2*j) + 1] = arm_cos_q31(aAnglesLib[j]);
284             }
285             EventRecord2(1,i,2);
286           }
```

Hands-On: CORDIC

- Click the “Build”



- “Start Debug” Session

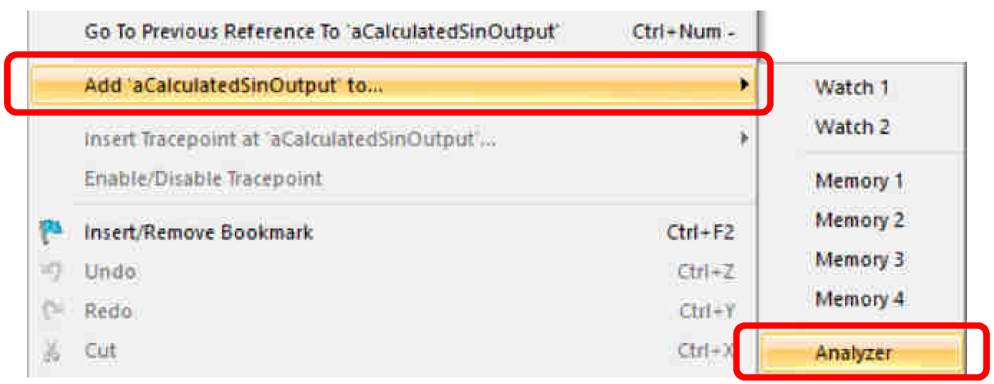


Enable Logic Analyzer

- On line 152, Right mouse click on “aCalculatedSinOutput” variable

```
151  /* Logic Analyzer variables */
152  static int32_t aCalculatedSinOutput = 0;
153  static int32_t aCalculatedCosOutput = 0;
154  int32_t aResults_ptr = 0;
```

- Move mouse over “Add ‘aCalculatedSinOutput’ to…”
- Click on “Analyzer”



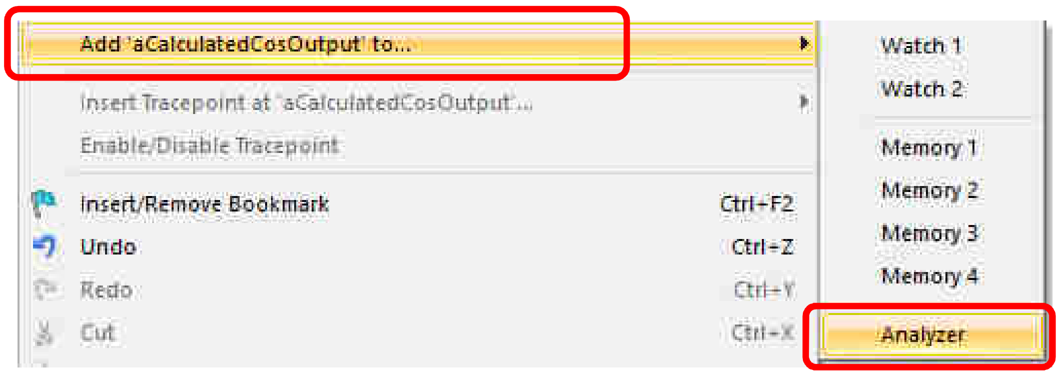
life.augmented

Enable Logic Analyzer

- On line 153, Right mouse click on “aCalculatedCosOutput” variable

```
151  /* Logic Analyzer variables */
152  static int32_t aCalculatedSinOutput = 0;
153  static int32_t aCalculatedCosOutput = 0;
154  int32_t aResults_ptr = 0;
```

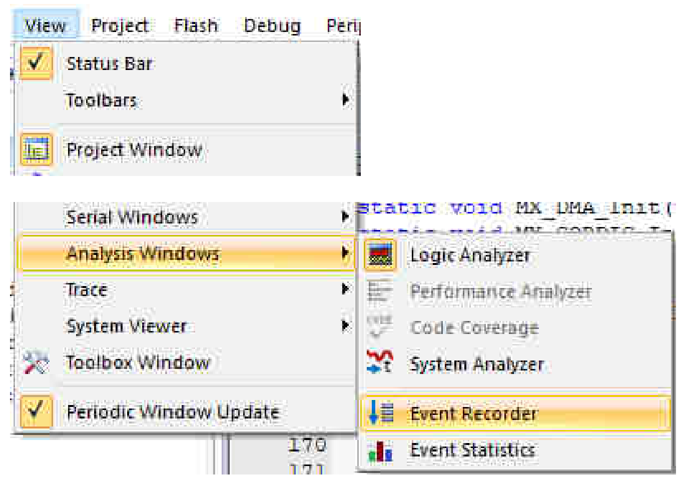
- Move mouse over “Add ‘aCalculatedCosOutput’ to…”
- Click on “Analyzer”

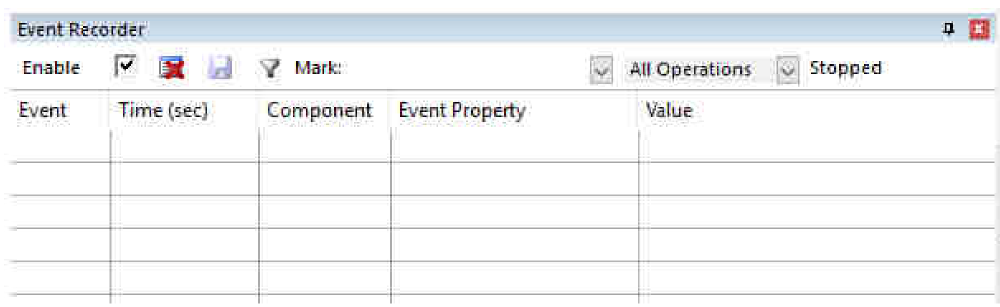


life.augmented

View Event Recorder

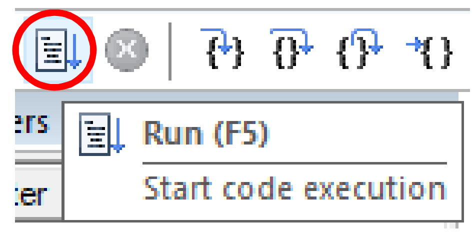
- Go to “View” menu
- Select “Analysis Windows”
- Select “Event Recorder”



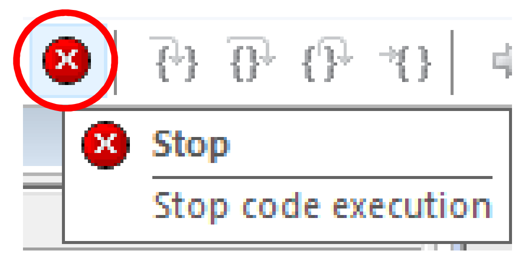


Run the Project

- “Run” the project
  - Capture trace data
  - Press “JOYSTICK_SEL” triggers “CORDIC” or “ARM Math library”
  - Green Led blinking
    - Fast blink (CORDIC used)
    - Slow blink (ARM Math Lib. Used)

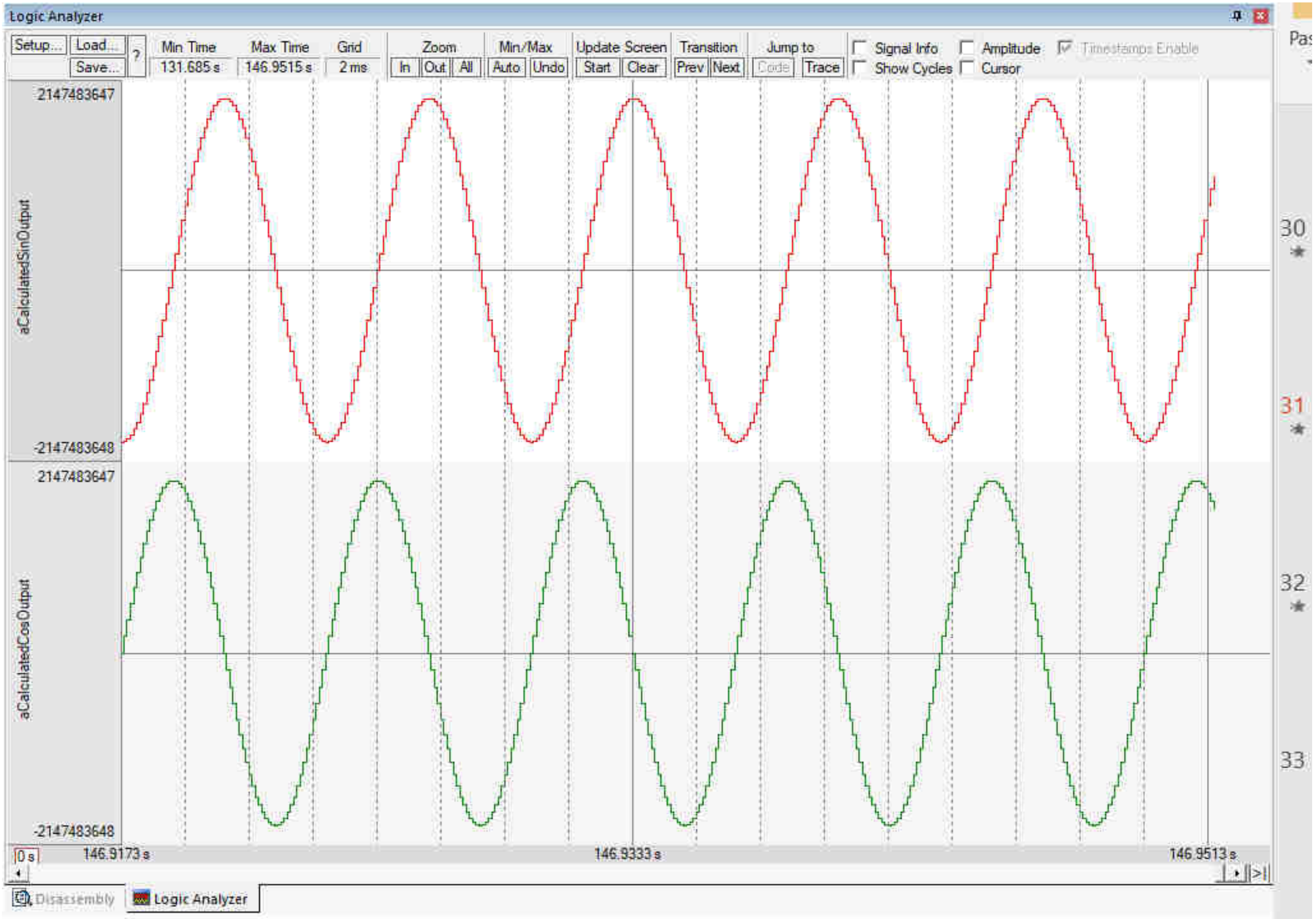


- “Stop” the project execution



Logic Analyzer

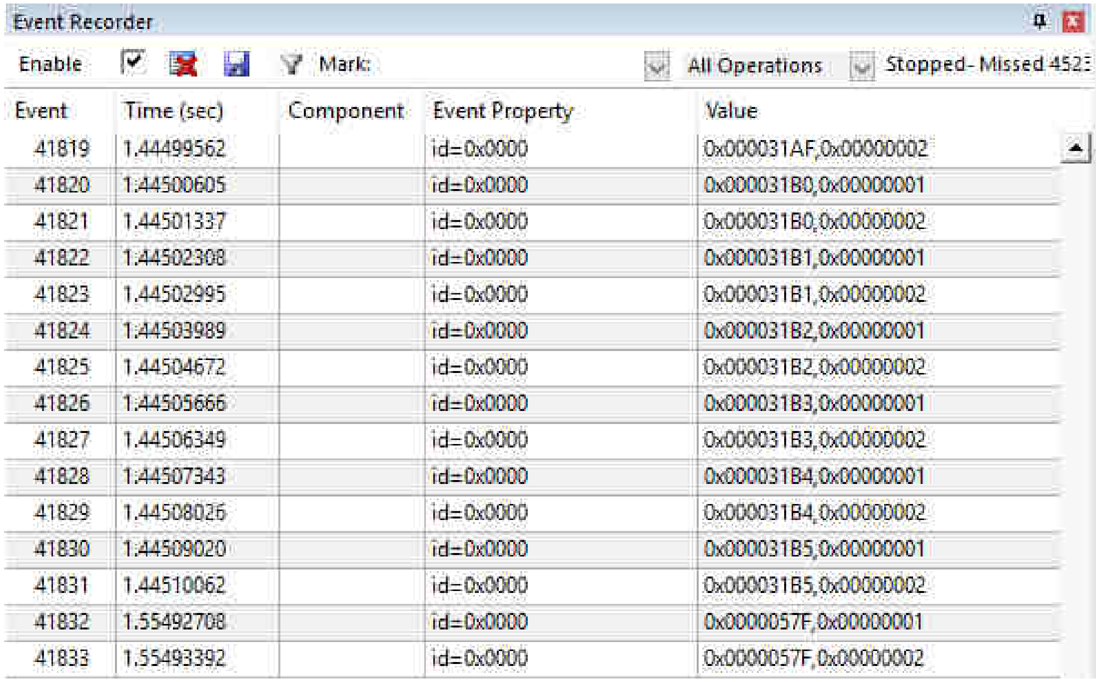
- CORDIC results are shown in Logic Analyzer



life.augmented

Event Recorder

- Events Captured
  - Event Number
  - Event Time Stamp
  - Event Property
    - Id = 0 (CORDIC)
    - Id = 1 (ARM Math Library)
  - Event Values
    - 1st: Loop Number
    - 2nd: Start(0x01) and End(0x02) of array calculation



Event Recorder

Enable | Mark: | All Operations | Stopped- Missed 4523

Event	Time (sec)	Component	Event Property	Value
41819	1.44499562		id=0x0000	0x000031AF,0x00000002
41820	1.44500605		id=0x0000	0x000031B0,0x00000001
41821	1.44501337		id=0x0000	0x000031B0,0x00000002
41822	1.44502308		id=0x0000	0x000031B1,0x00000001
41823	1.44502995		id=0x0000	0x000031B1,0x00000002
41824	1.44503989		id=0x0000	0x000031B2,0x00000001
41825	1.44504672		id=0x0000	0x000031B2,0x00000002
41826	1.44505666		id=0x0000	0x000031B3,0x00000001
41827	1.44506349		id=0x0000	0x000031B3,0x00000002
41828	1.44507343		id=0x0000	0x000031B4,0x00000001
41829	1.44508026		id=0x0000	0x000031B4,0x00000002
41830	1.44509020		id=0x0000	0x000031B5,0x00000001
41831	1.44510062		id=0x0000	0x000031B5,0x00000002
41832	1.55492708		id=0x0000	0x0000057F,0x00000001
41833	1.55493392		id=0x0000	0x0000057F,0x00000002

	Execution Time
CORDIC	6.84 us
ARM Math Library	42.24 us

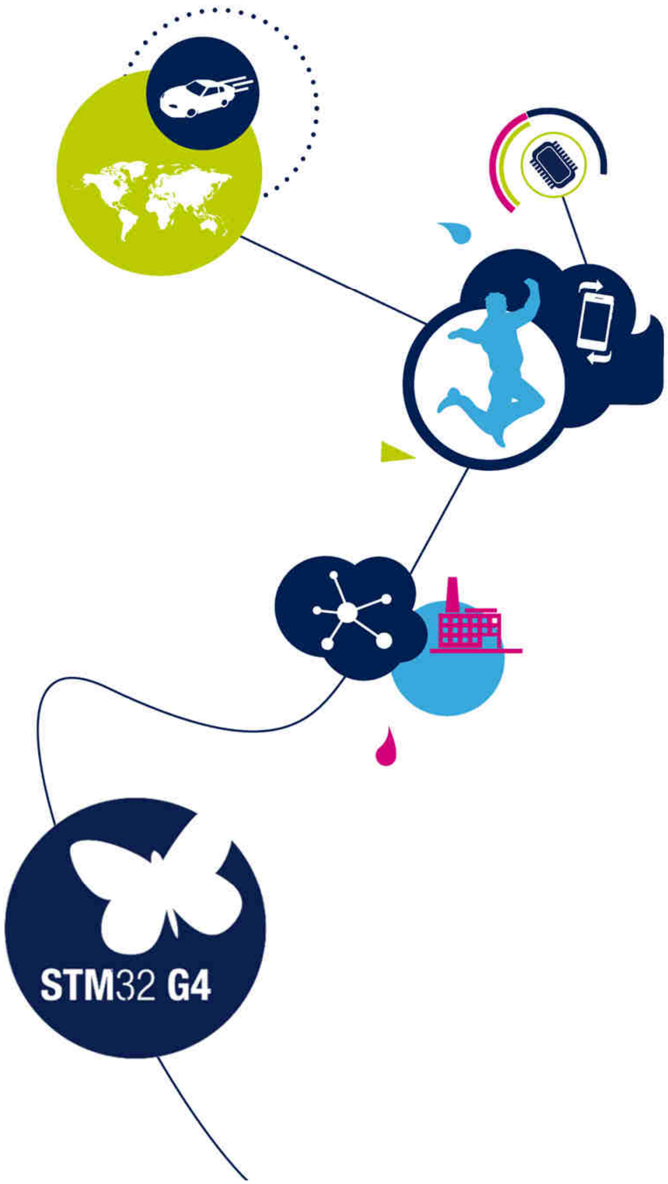
life.augmented

CORDIC: Additional Notes

- Use Cube Low Level driver to reduce HAL overhead
- AN5325: Getting started with the CORDIC accelerator using the STM32CubeG4 MCU Package
- Keil Application Note 321: Event Recorder Debugging with STM32G Cortex-M0/M0+

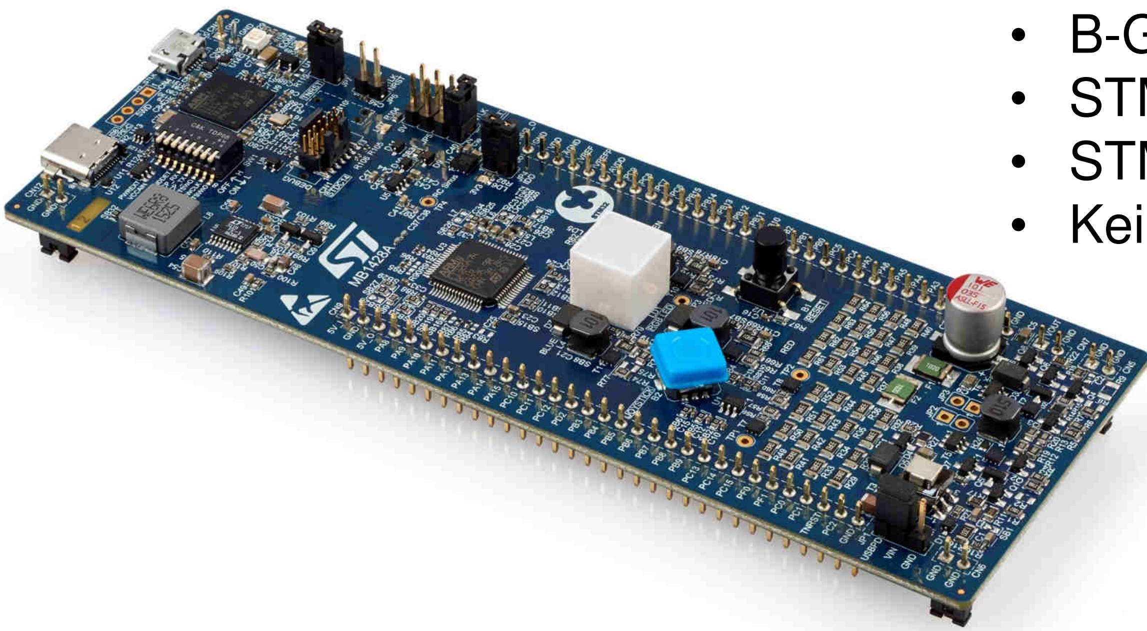
STM32G4 mixed-signal MCU workshop

Hands On:
Filter Math Acclerator (FMAC)



life.augmented

Systems Check



- USB cable
- B-G474E-DPOW1 Discovery Board
- STM32G4 Family Cube Library
- STM32CubeMx
- Keil MDK-ARM

Filter Math Accelerator

G4 FMAC

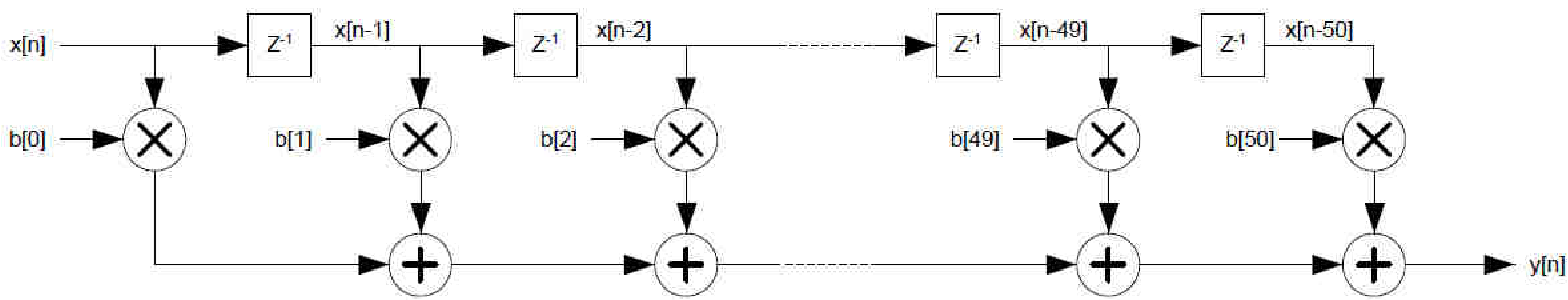
- Multiplier (16x16)
- Accumulator (24bit + 2bit)
- Address generation logic
- Input and Output Circular Buffers
- FIR & IIR filters possible
- Dot product, convolution, correlation
- Does not stall the CPU
- 256 x 16bit local memory
- Built-in sequencer
- Runs at the System Clock speed

Internally connected to:

- DMA
- AHB slave interface
- NVIC

G4 Adaptive FIR Filter

Figure 7. **51-tap FIR filter**



- FIR filter is a convolution of the signal with filter coefficients
- The number of coefficients determines the minimum notch size
- Calculating N coeffcients requires N*N operations

STM32 G4

Filter Math Accelerator

G4 Adaptive FIR Filter - Quantities

- Filter Coefficients will probably be in floating point format
- Coefficient conversion to 16 bit fixed point is necessary
- Fixed point format is Q1.15
    - Bits [0..14] represent fractional digits
    - Bit 15 is a sign bit
    - Range = -1.0 (0x8000) to +0.999969482421875 (0x7FFF)
- Proper design will keep the absolute value coeffiecients < or = 1
    - If not, all coefficients will need to be divided by the greatest coefficient
- To convert floating to fixed point, the Floating Point Unit is used
    - FPU takes 8 cycles per conversion
    - 51 coefficients take ~408 clock cycles

life.augmented

Lab steps 1 of 6

This Hands On Lab illustrates noise cancellation using an adaptive FIR filter.

The experiment will be run on two input frames:

- a gaussian noise with two tone interferers.

- a gaussian noise with three tone interferers.

Create individual coefficient sets.

The result is a "whitened" noise.

Figure 1. Gaussian noise with two-tone interferer

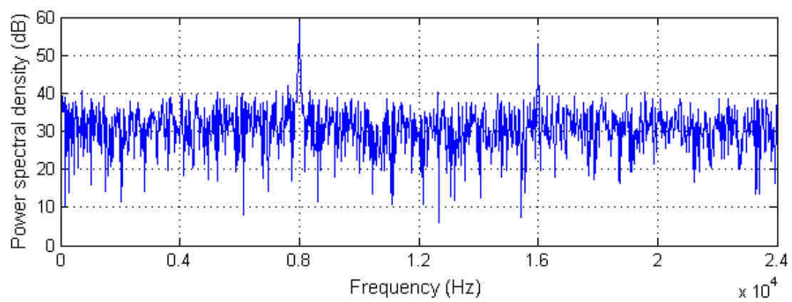
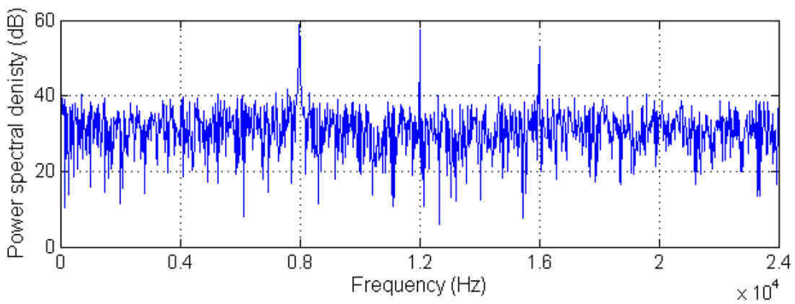


Figure 4. Gaussian noise with three-tone interferer



Filter Math Accelerator

Lab steps 2 of 6

- The FMAC peripheral will be configured to perform an adaptive FIR filter.
- There will be 51 feed-forward taps.
- The gain is equal to 0.
- The input and output thresholds are set to 1.
- The clipping feature can be enabled or disabled thanks to the compilation switch CLIP_ENABLED in main.h.

Filter Math Accelerator

Lab steps 3 of 6

The FMAC peripheral is making the best use of its 256 x 16-bit internal memory.

The local memory is divided into three buffers:

- 0 to 50: coefficient buffer;
- 51 to 150: input buffer;
- 151 to 250: output buffer;
- 251 to 255: unused.

Filter Math Accelerator

Lab steps 4 of 6

- DMA mode is used to transfer input data from memory TO the FMAC peripheral.

- DMA mode is used to transfer output data FROM FMAC peripheral to memory.

- The coefficient buffer contains the vector of feed-forward taps
- This is named the B buffer
  - (size comprised in [2:127]; here, size N = 51 elements).

- This buffer is initialized during the configuration step in DMA mode.

Filter Math Accelerator

Lab steps 5 of 6

The size of the input buffer is the sum of:

- the size of the coefficient vector B (N = 51 elements);
- the size of the additional space D1 needed for throughput improvement (minimum: 0; here, D1 = 49 elements).

The input buffer is filled during the preload step in DMA mode;
new values are added in DMA mode once the preloaded ones have been consumed.

The size of the output buffer is the one of the additional space D2 needed for a better throughput (minimum: 1; here, D2 = 100 elements).
The output buffer is read in DMA mode.

Lab steps 6 of 6

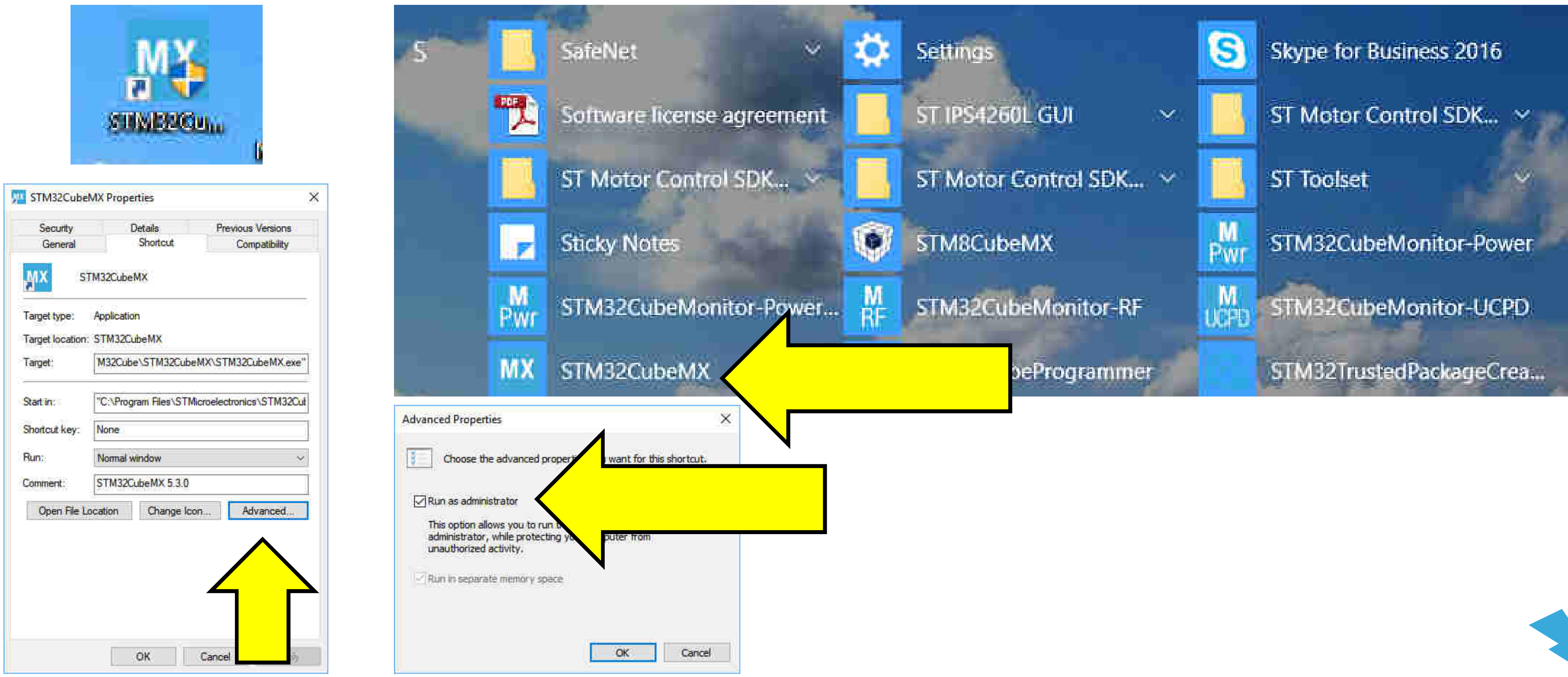
STM32 board LEDs are used to monitor the example status:

- LED4 is ON when correct FMAC FIR results have been calculated.
- LED5 is blinking (1 second period) if an error is detected in the FIR filter results or if there is an initialization or configuration error.

The filtered output is displayed in the Terminal I/O if the compilation switch PRINT_OUTPUT is enabled in main.h.

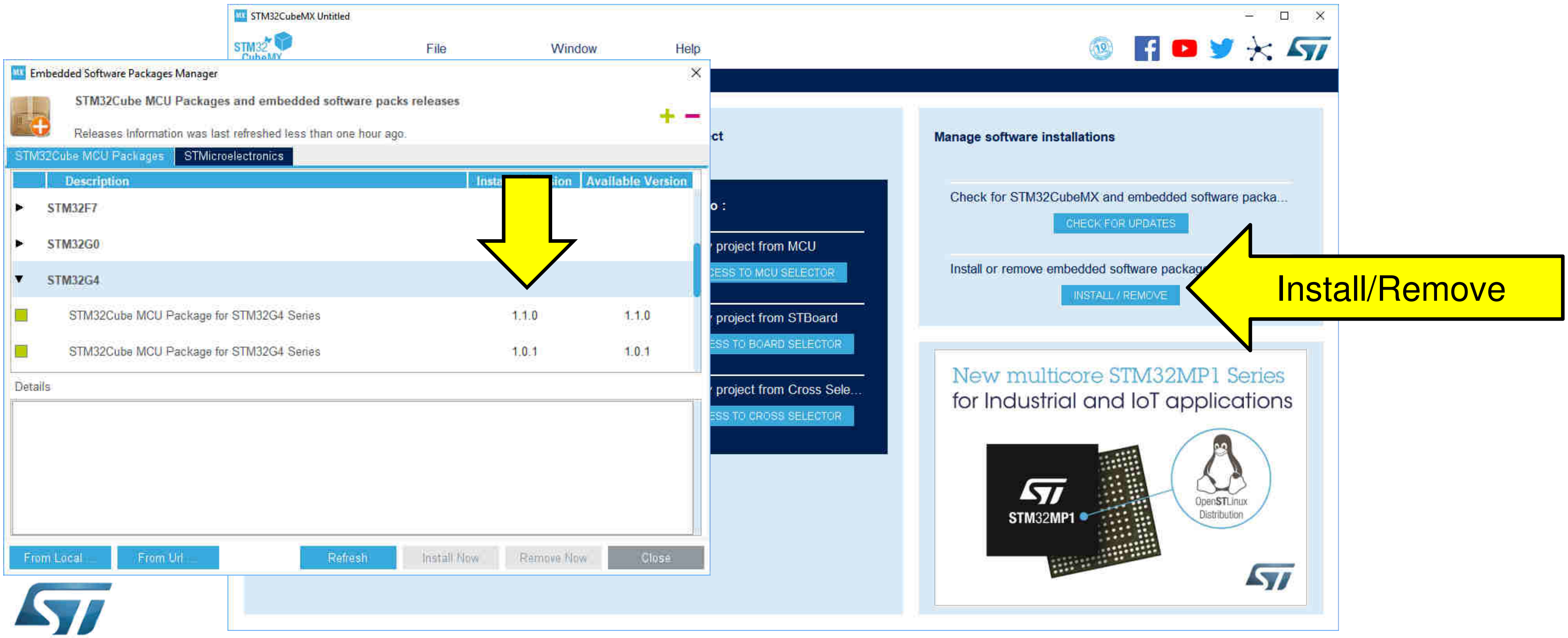
STM32CubeMx

Start up

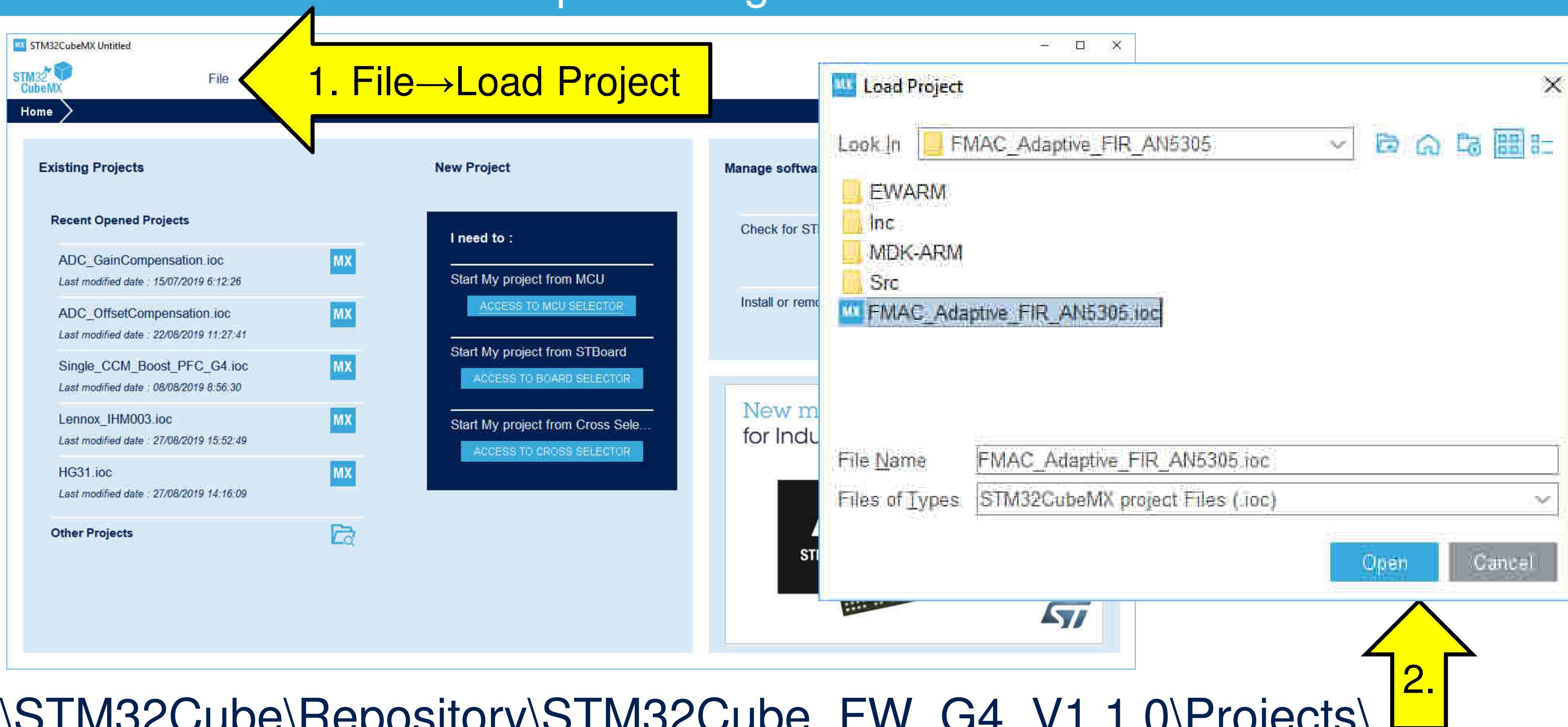


STM32CubeMx

Splash Page – G4 Library Installed?



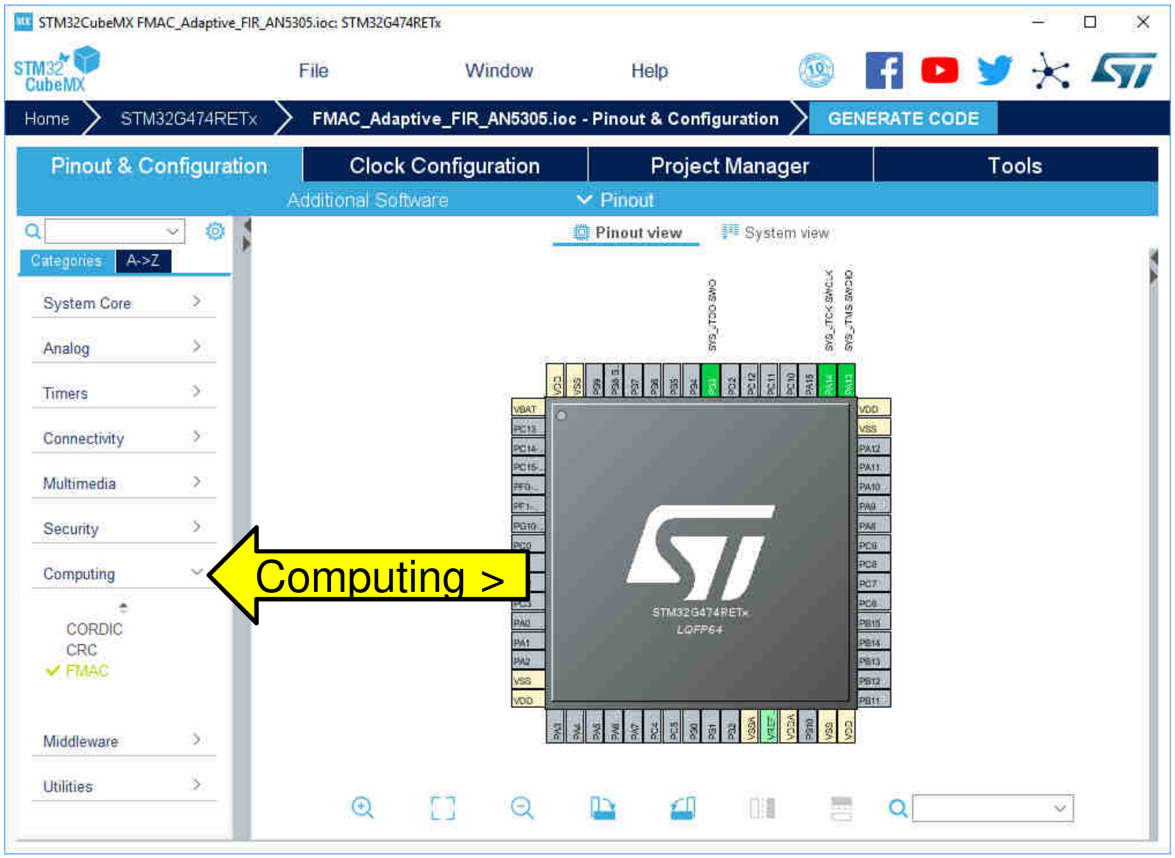
Splash Page



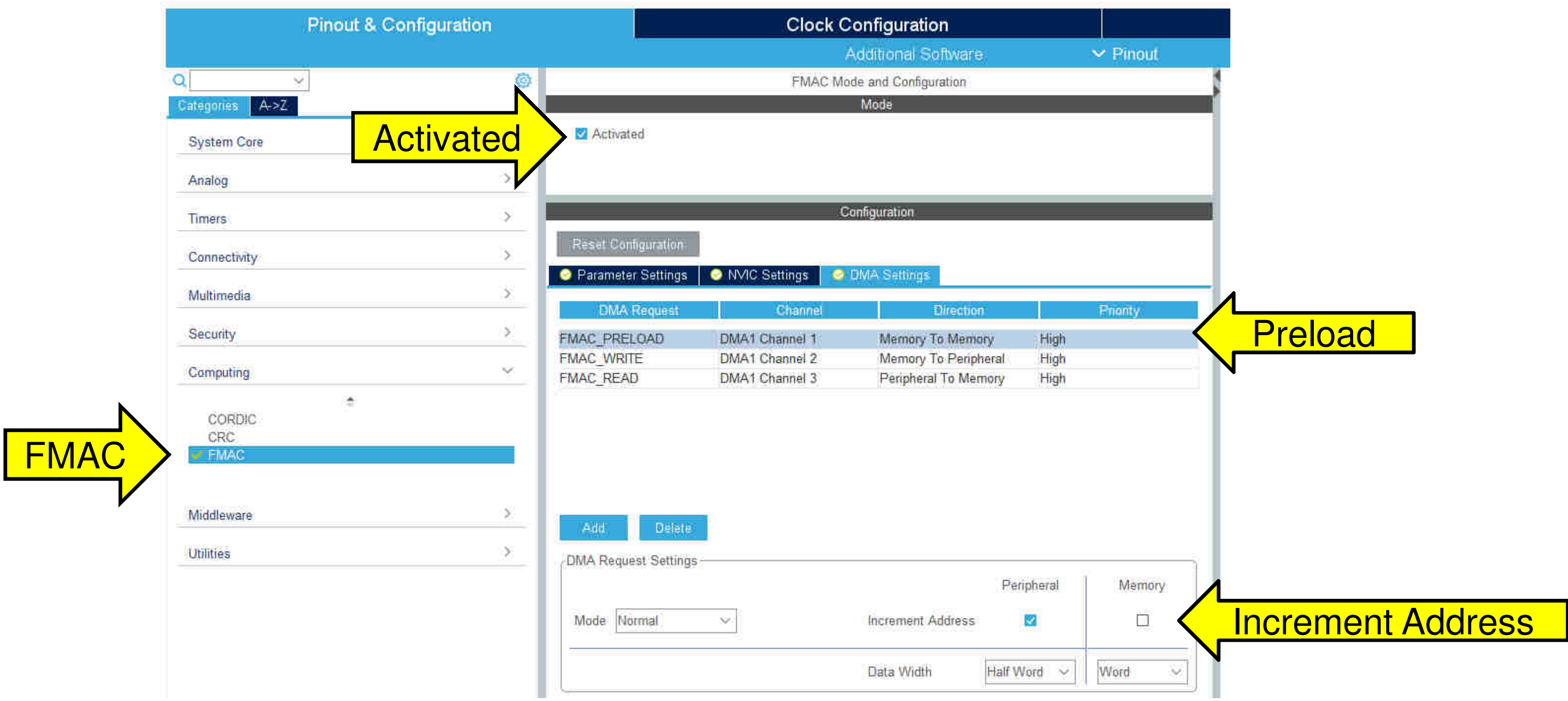
C:\...\STM32Cube\Repository\STM32Cube_FW_G4_V1.1.0\Projects\
B-G474E-DPOW1\Examples\FMAC\FMAC_Adaptive_FIR_AN5305

STM32CubeMx

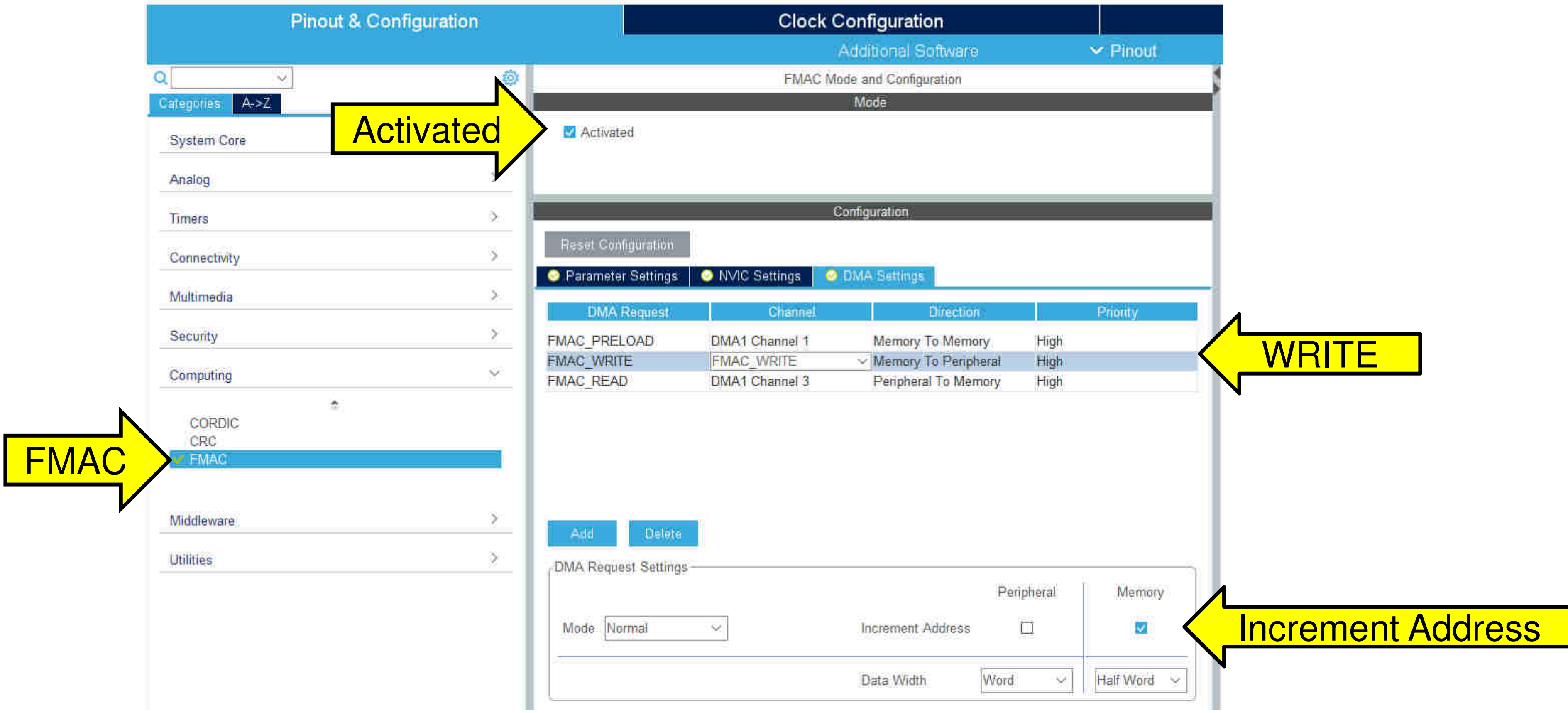
Graphic Design Representation



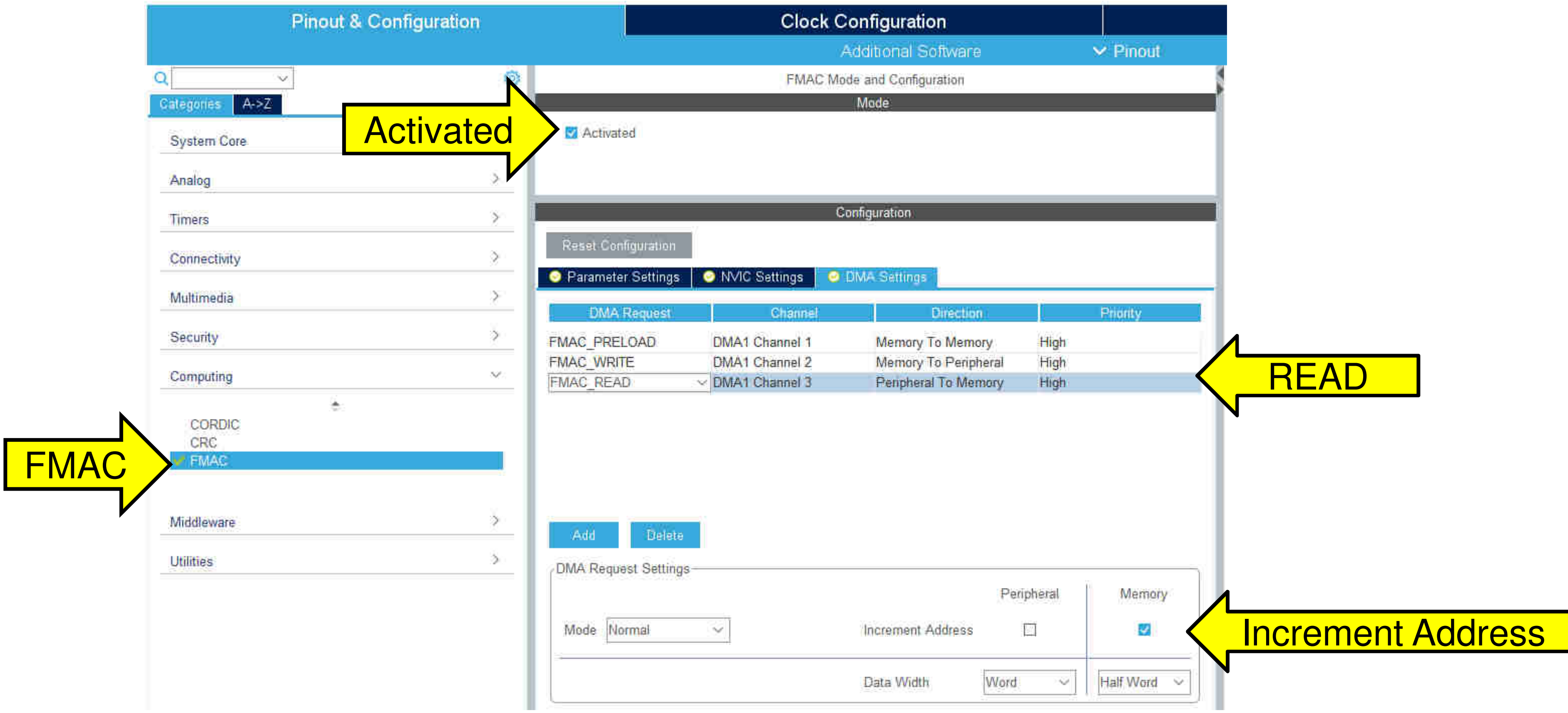
FMAC – 3 DMA Channels



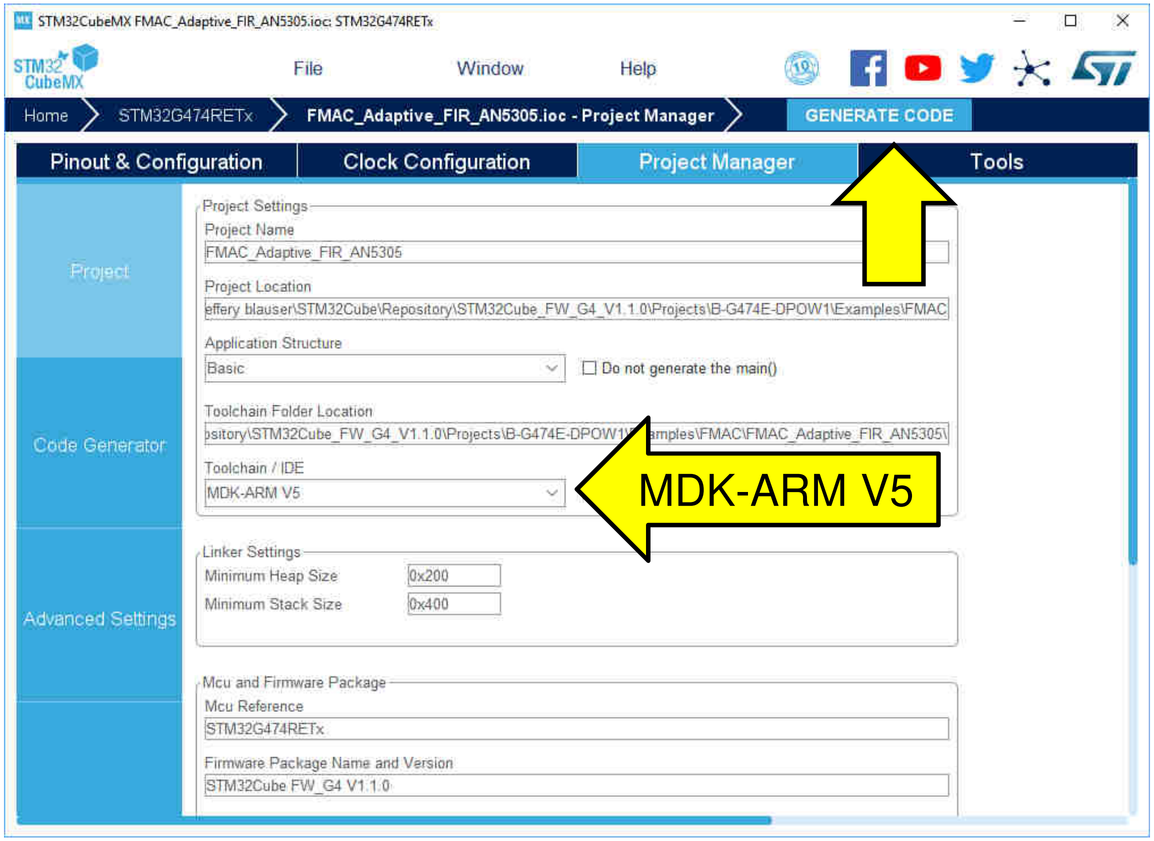
FMAC – 3 DMA Channels



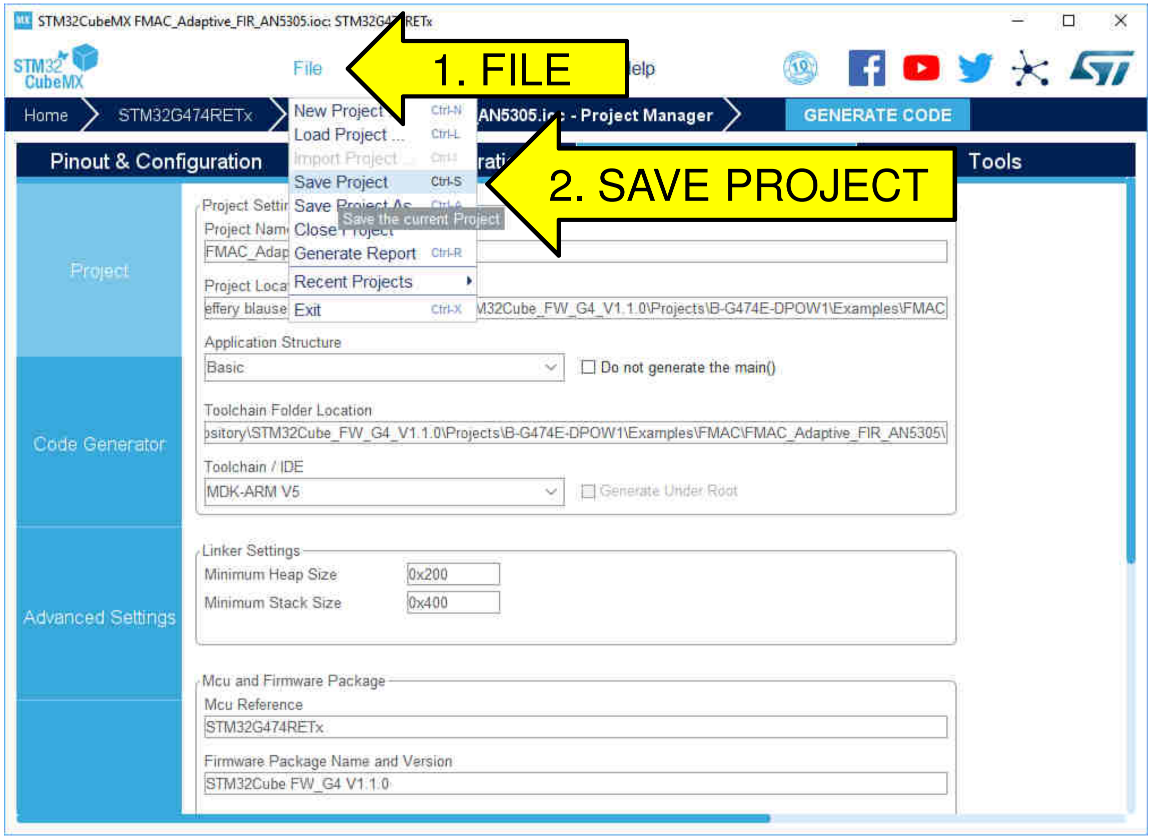
FMAC – 3 DMA Channels



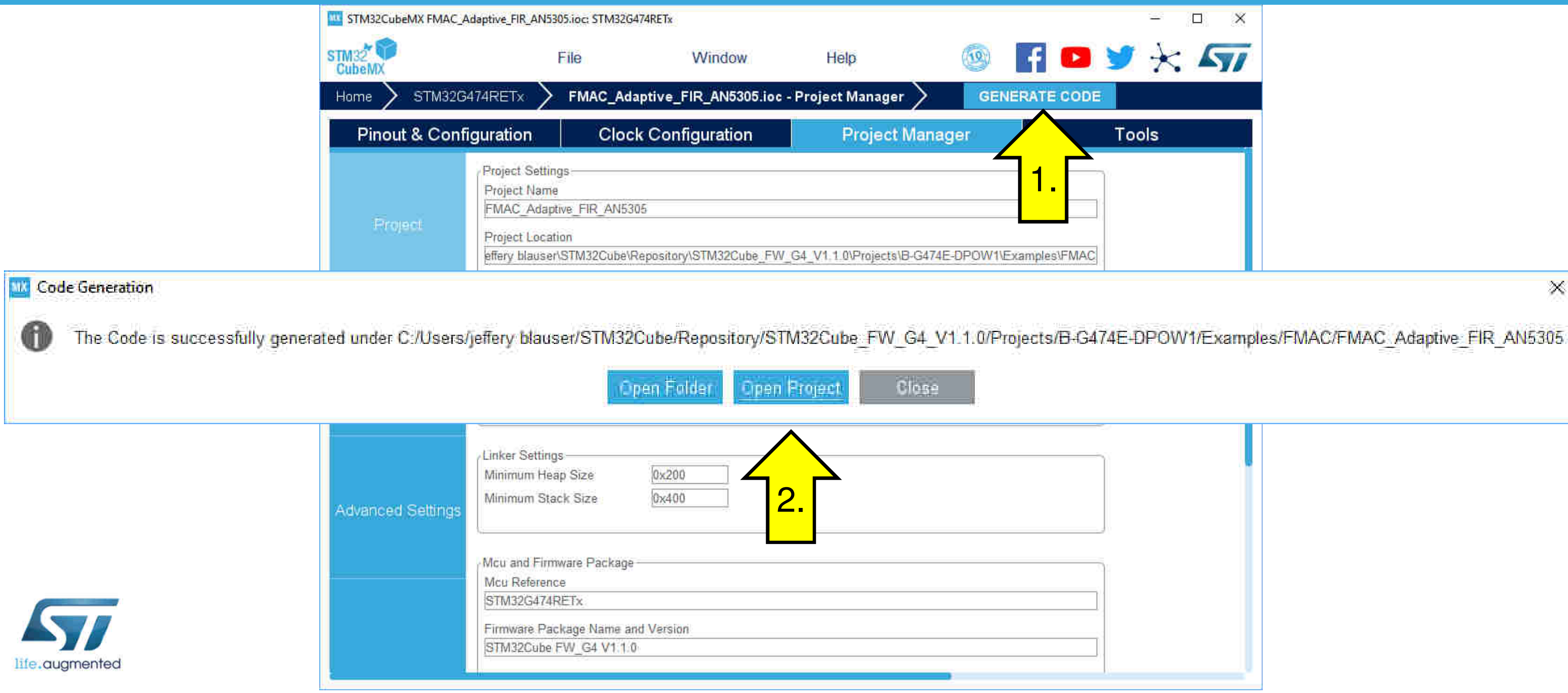
Project Manager – Keil MDK-ARM V5



FILE→SAVE PROJECT

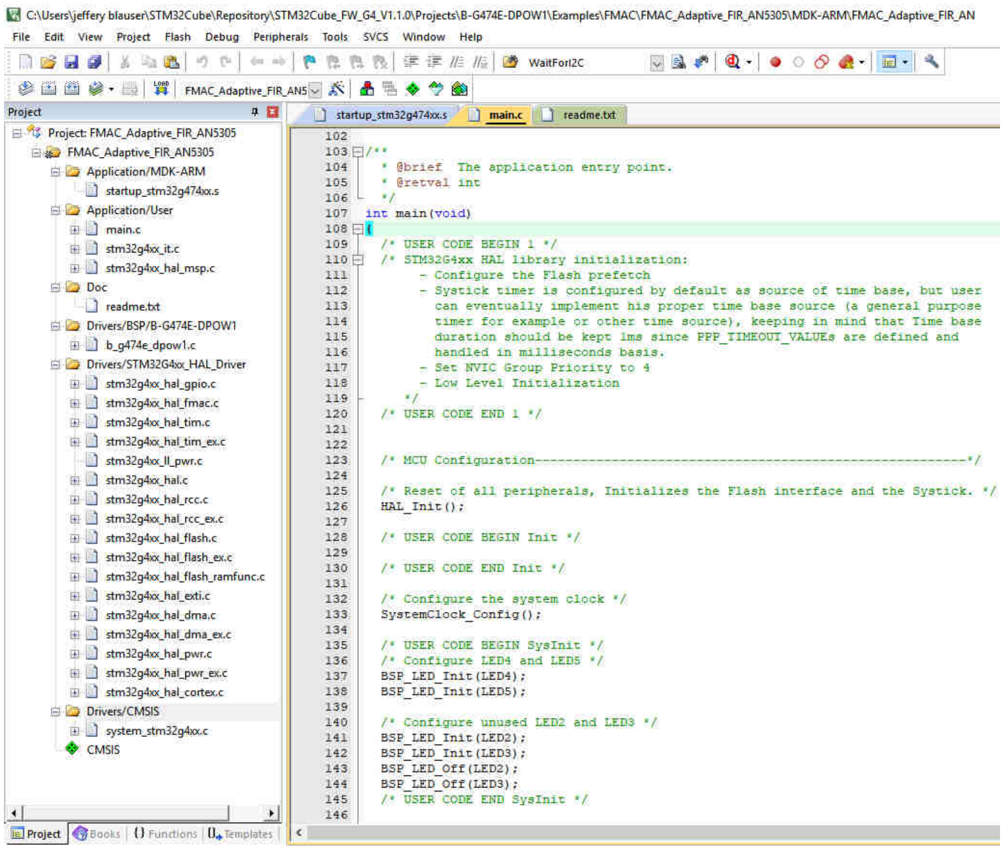


Generate Code

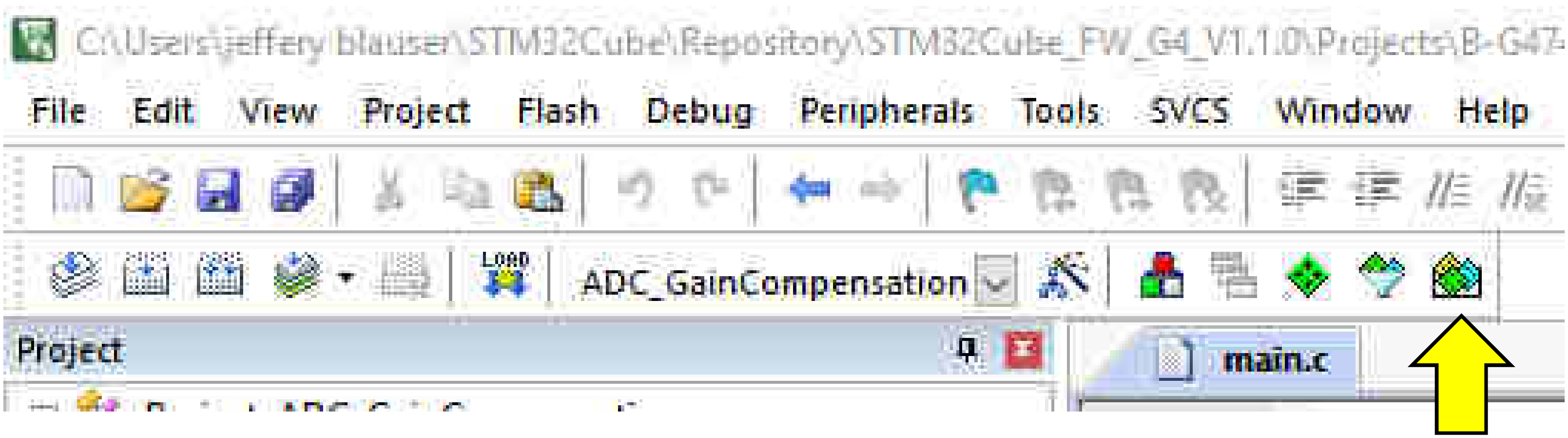


life.augmented

Keil Start up page

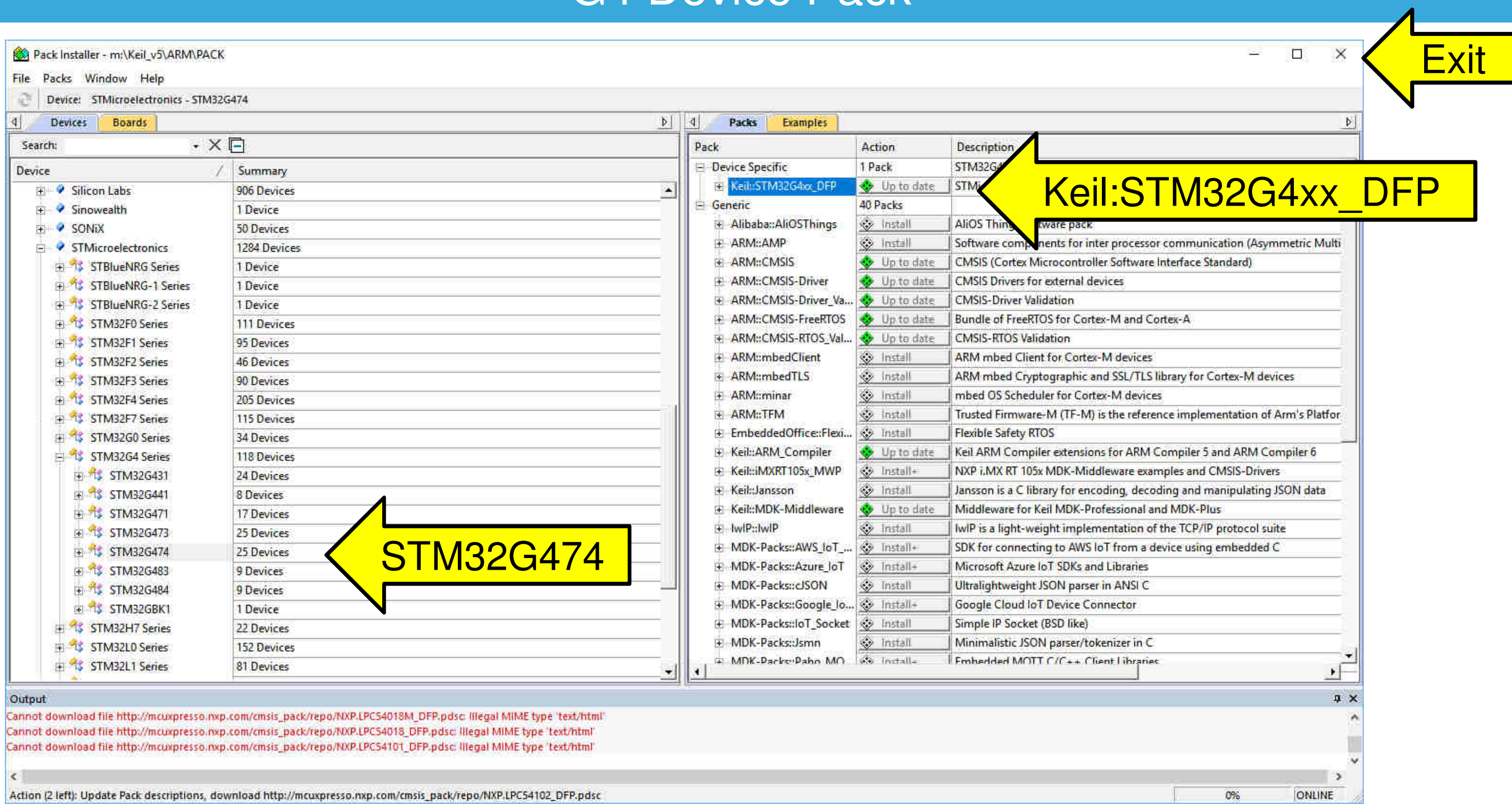


Check the Device Pack

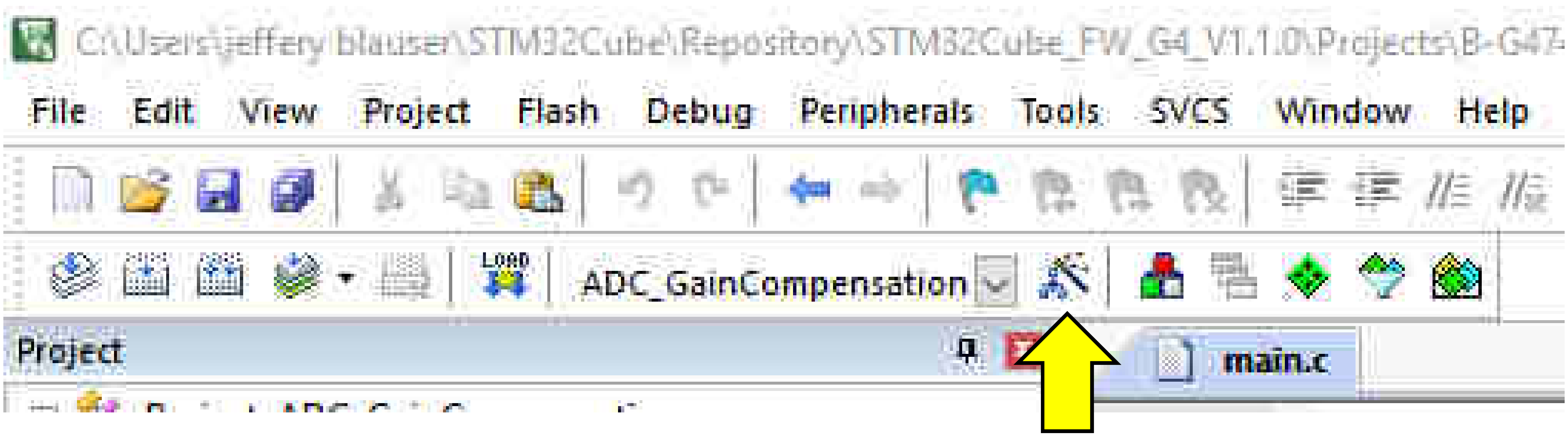


Keil MDK-ARM

G4 Device Pack

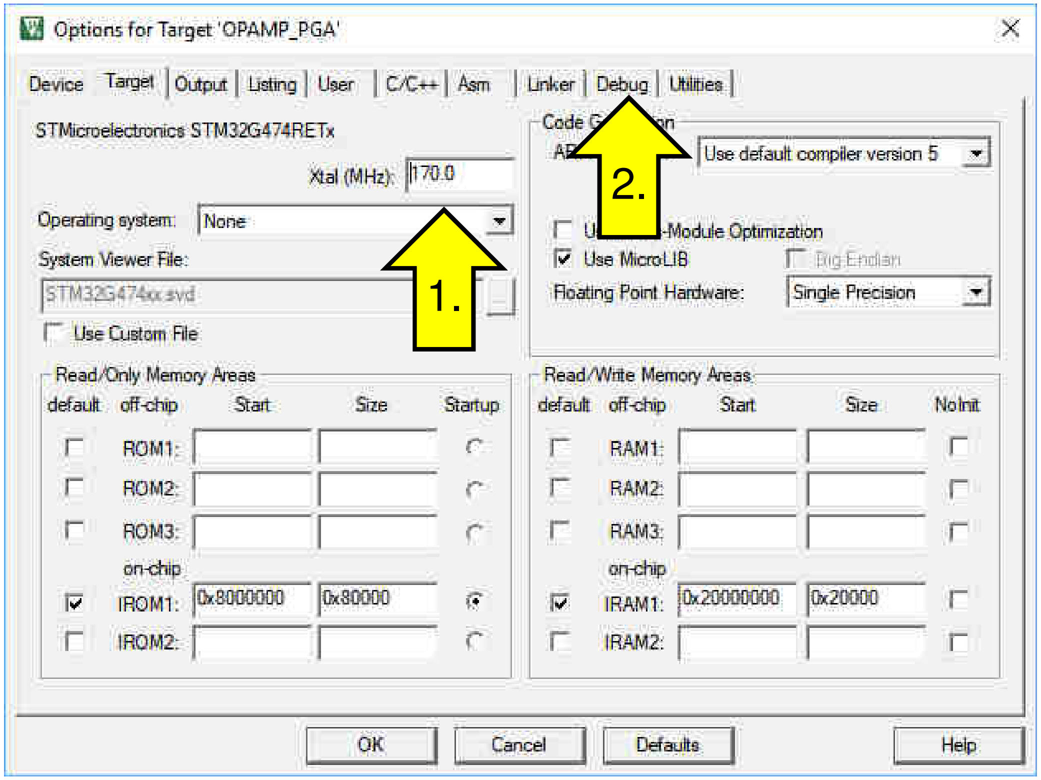


Check the Options

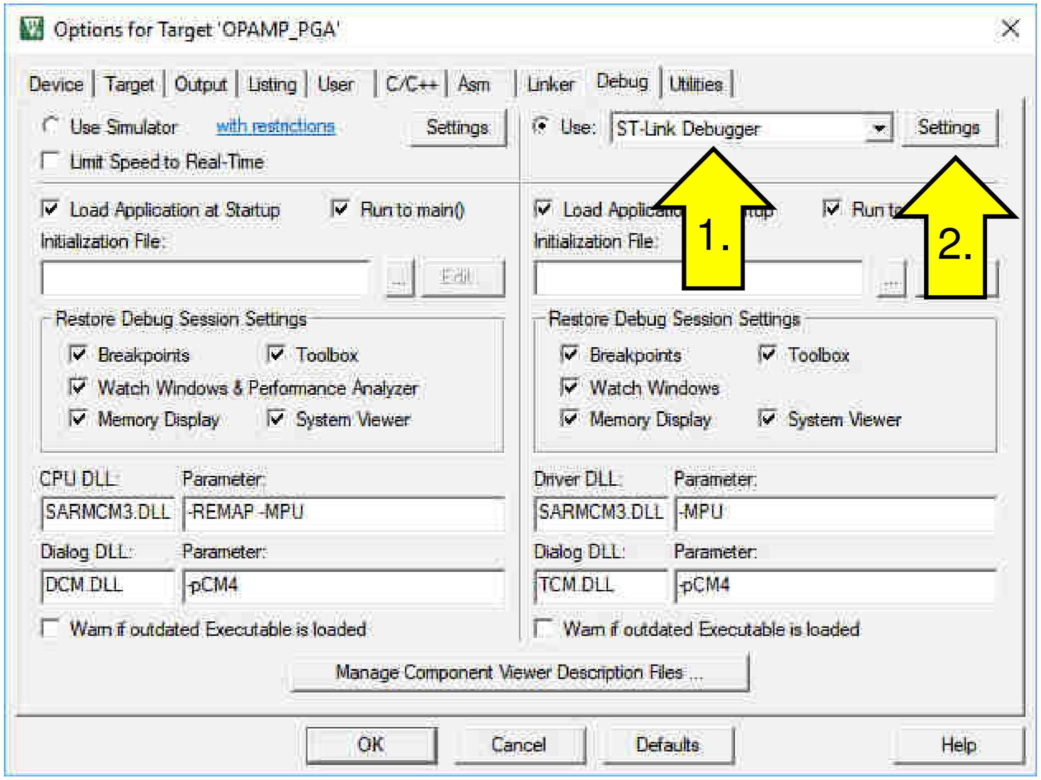


Keil MDK-ARM

Clock speed and Debug

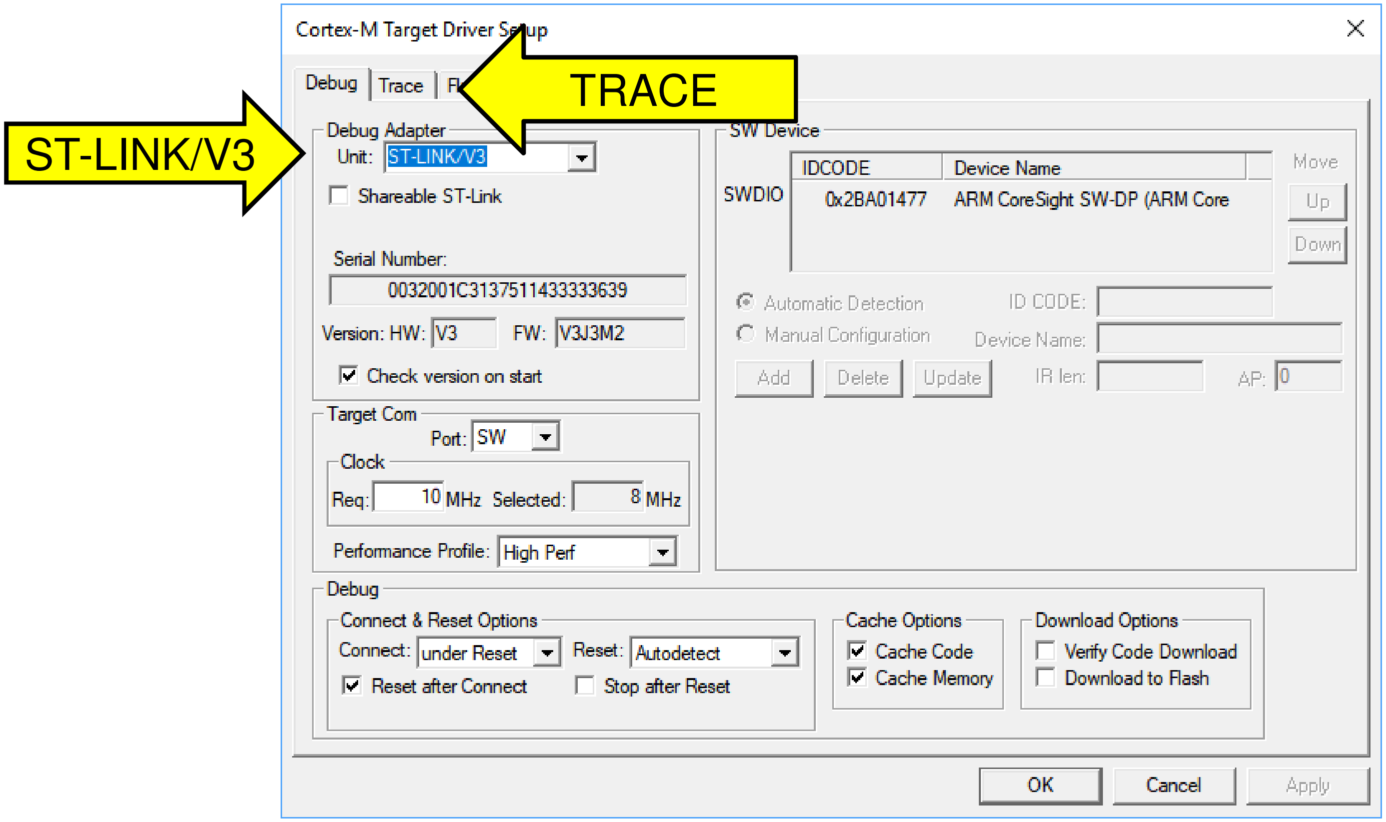


ST-Link Debugger - Settings



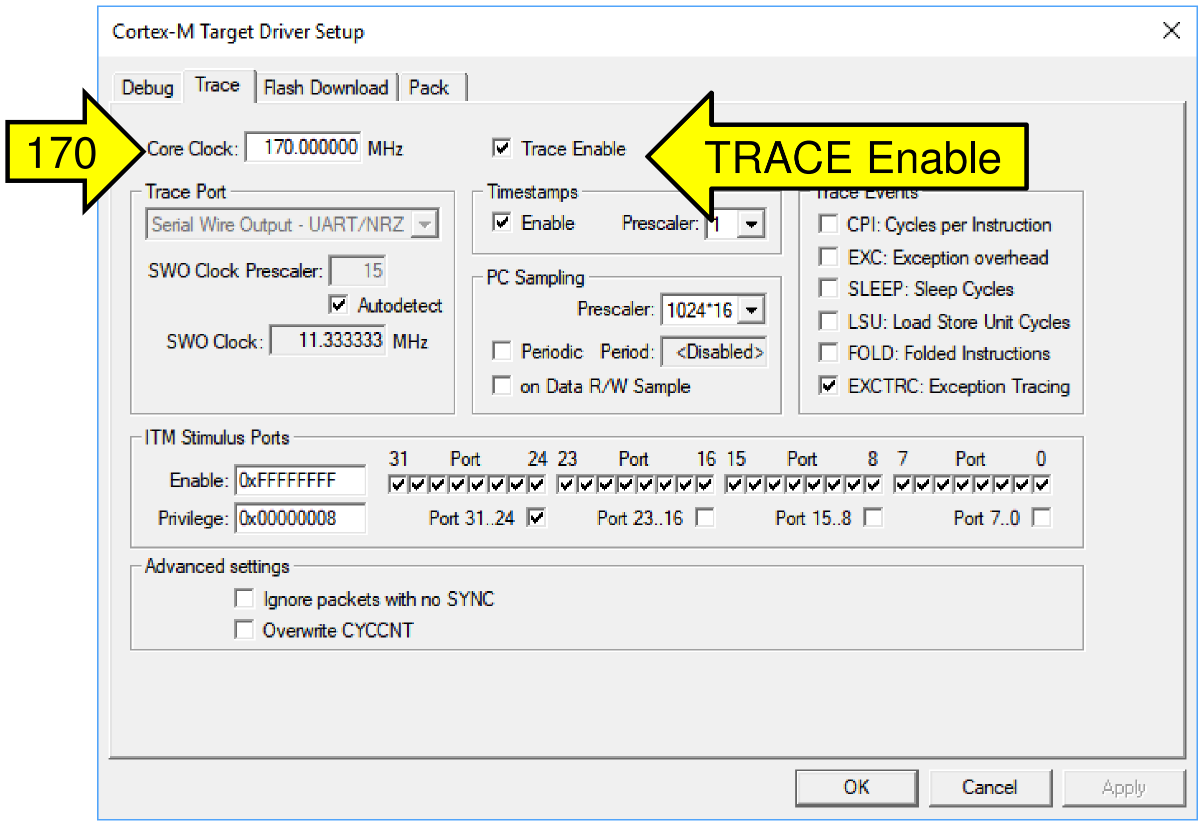
Keil MDK-ARM

ST-Link/V3 & TRACE



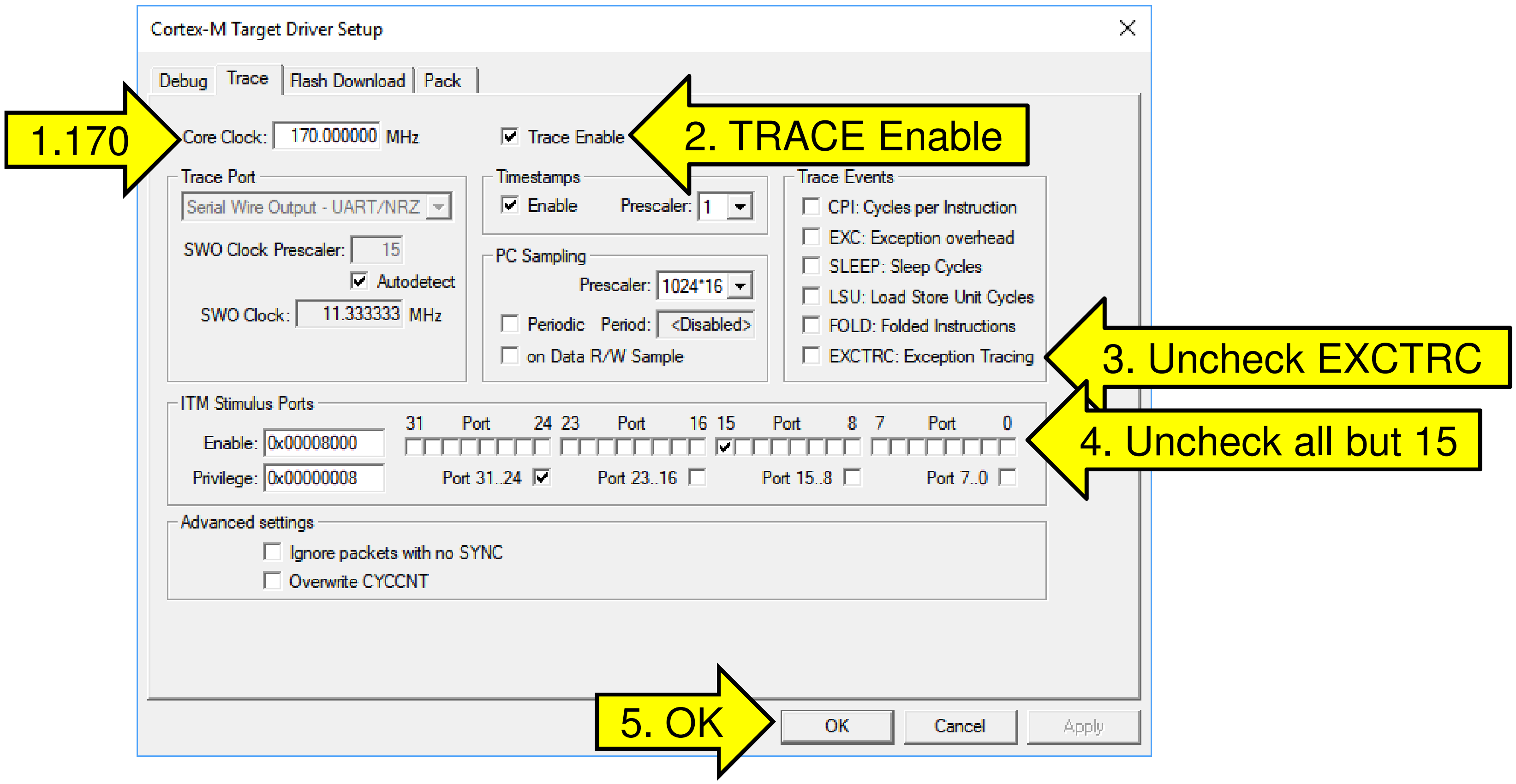
Keil MDK-ARM

Core Clock & TRACE Enable (Before)

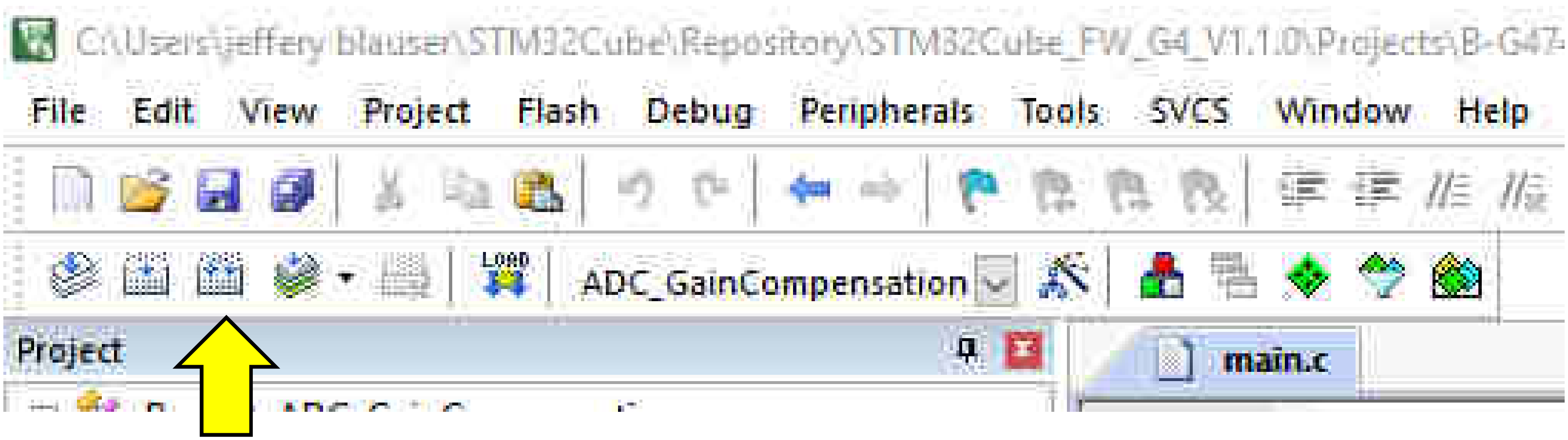


Keil MDK-ARM

Core Clock & TRACE Enable (After)



Build

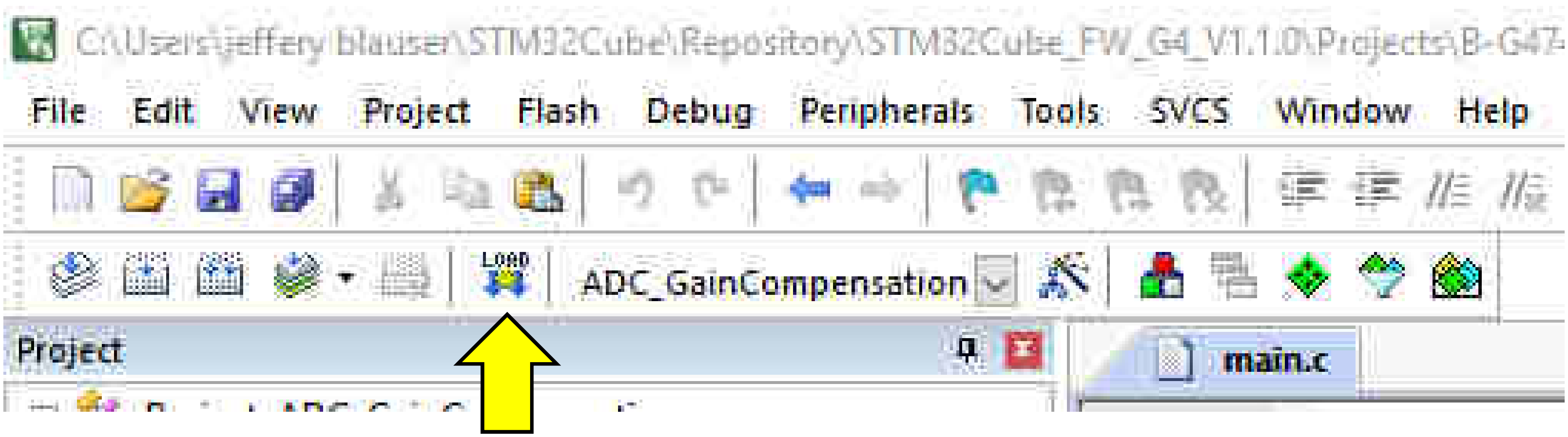


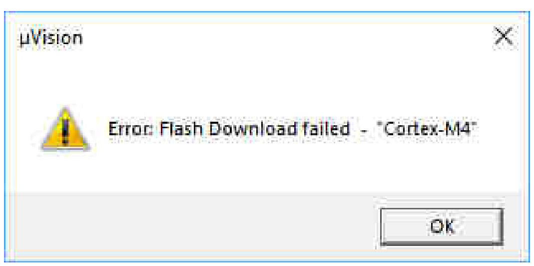
Build Output

```
Program Size: Code=7876 RO-data=8972 RW-data=116 ZI-data=13804
FromELF: creating hex file...
FMAC_Adaptive_FIR_AN5305\FMAC_Adaptive_FIR_AN5305.hex: Warning: Q9931W: Your license for feature Keil will expire in 29 days
Finished: 0 information, 1 warning and 0 error messages.
"FMAC_Adaptive_FIR_AN5305\FMAC_Adaptive_FIR_AN5305.axf" - 0 Error(s), 0 Warning(s).
Build Time Elapsed:  00:01:00
```

Keil MDK-ARM

Load





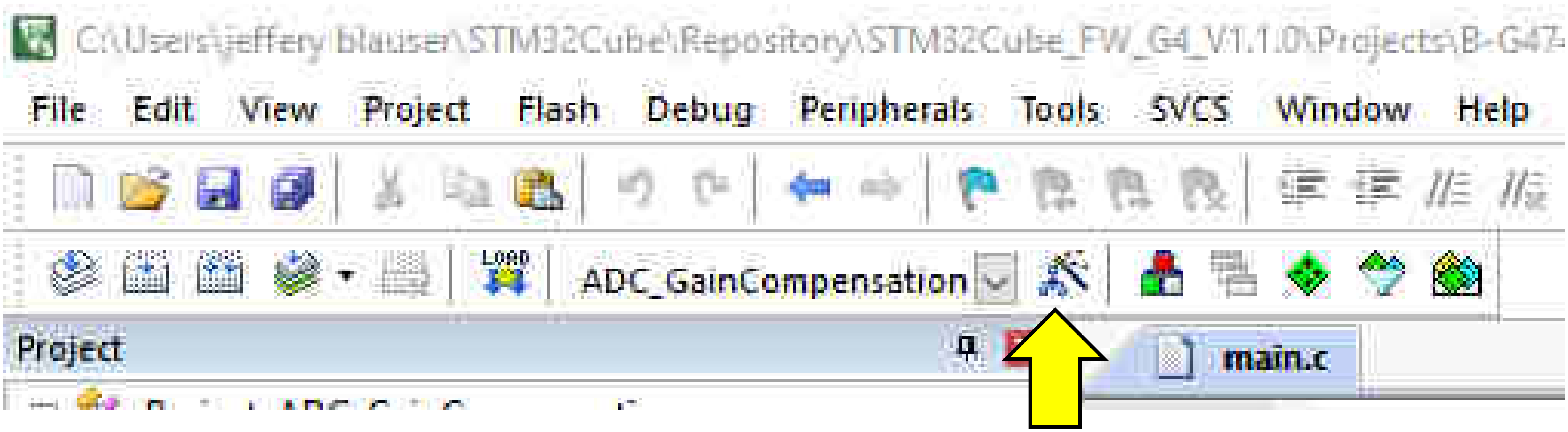
Build Output

```
Contents mismatch at: 08001860H  (Flash=38H  Required=C5H) !
Contents mismatch at: 08001861H  (Flash=46H  Required=F8H) !
Contents mismatch at: 08001862H  (Flash=FEH  Required=88H) !
Contents mismatch at: 08001863H  (Flash=F7H  Required=00H) !
Contents mismatch at: 08001864H  (Flash=B9H  Required=60H) !
Contents mismatch at: 08001865H  (Flash=FCH  Required=6CH) !
Too many errors to display !
Error: Flash Download failed  -  "Cortex-M4"
```

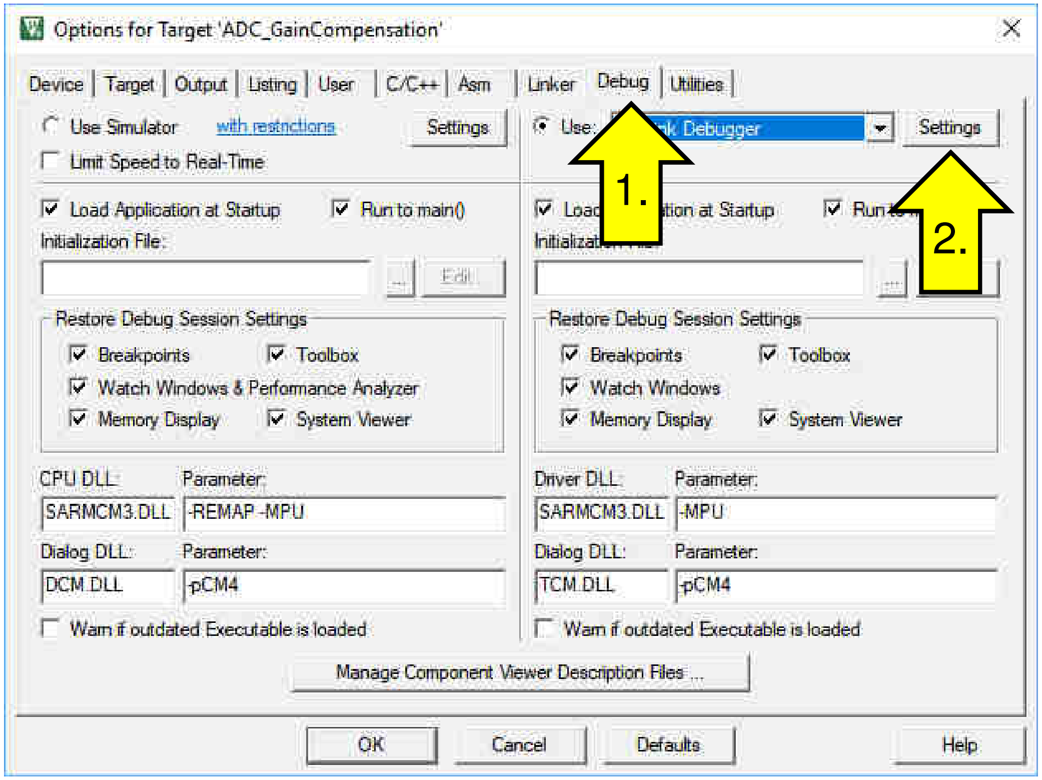
life.augmented

Keil MDK-ARM

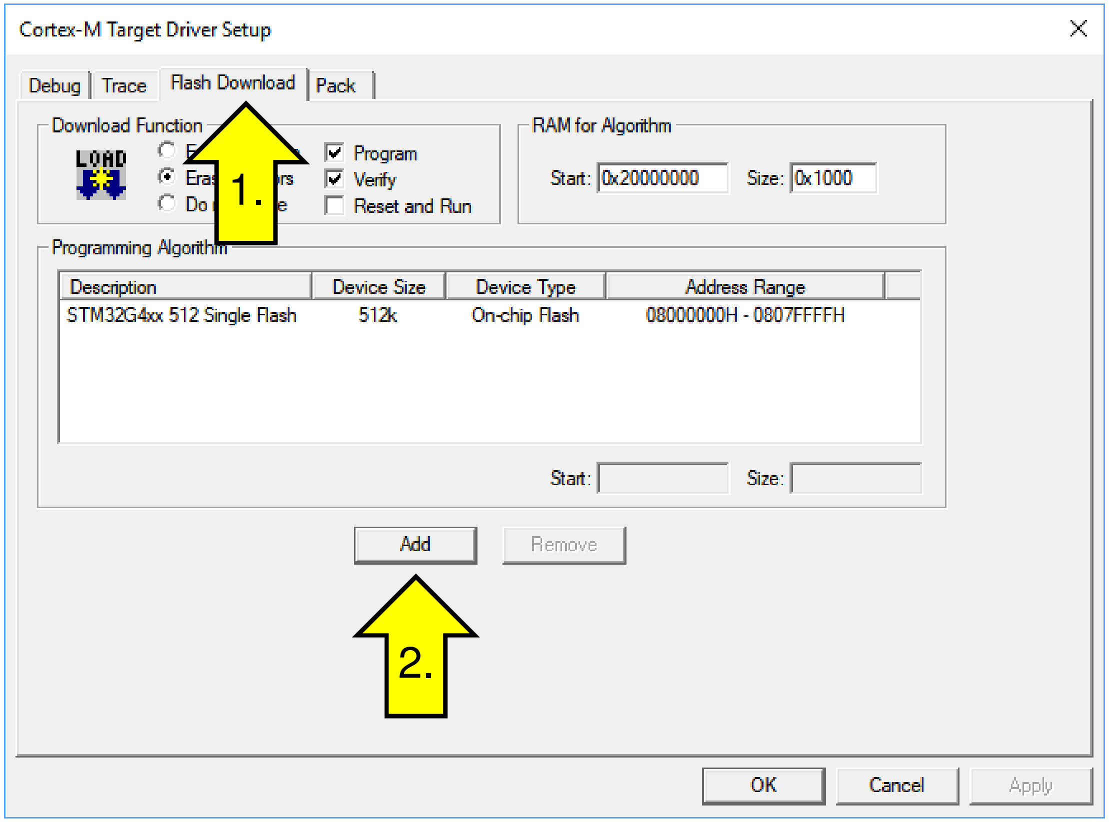
Options



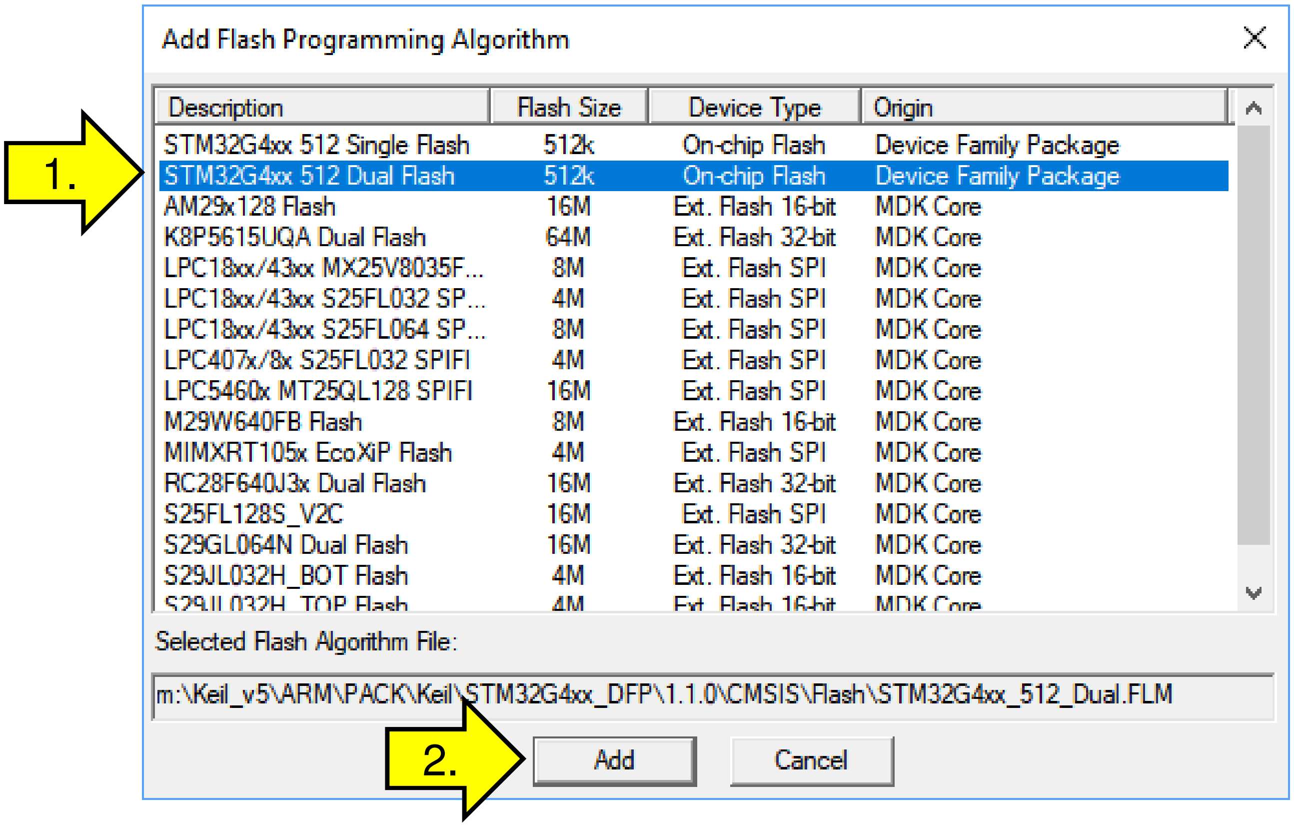
ST-Link Debugger - Settings



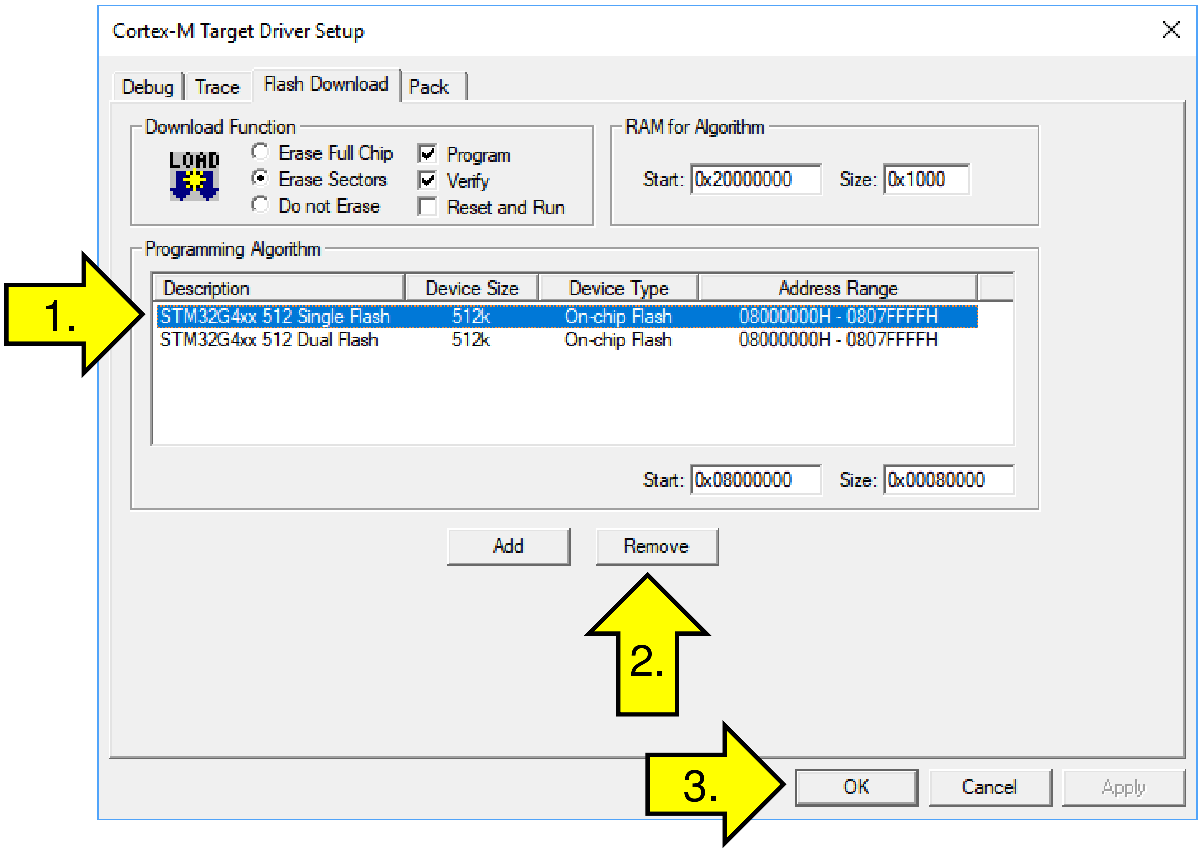
Flash Download → Add



STM32G4xx 512 Dual Flash

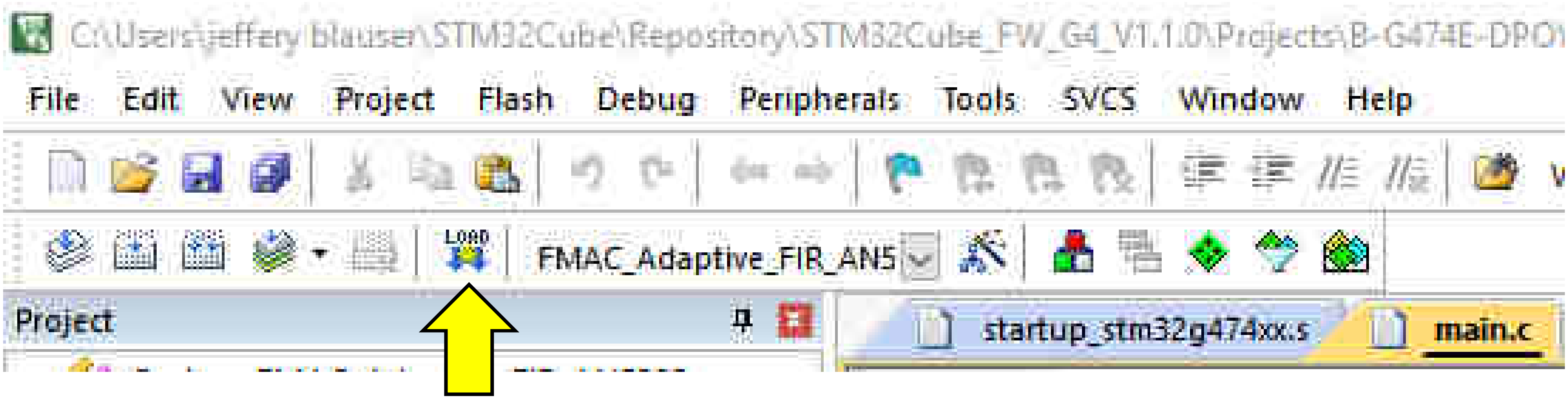


Remove Single Flash



Keil MDK-ARM

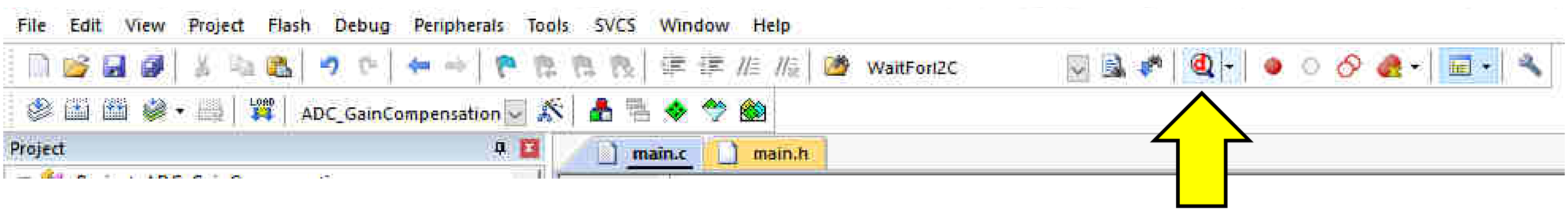
Load



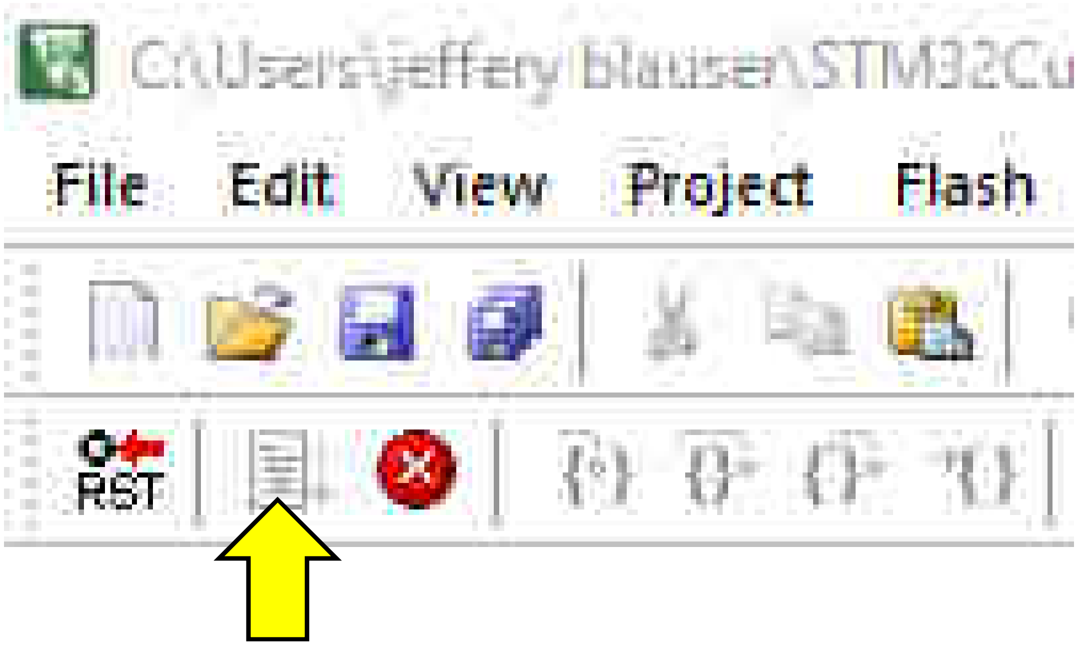
Build Output

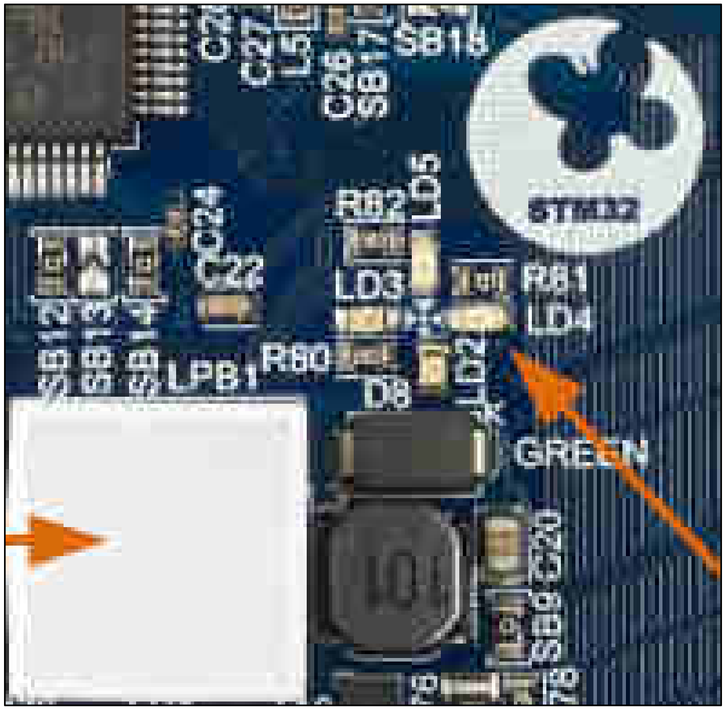
```
Build Time Elapsed:  00:01:00
Load "FMAC_Adaptive_FIR_AN5305\\FMAC_Adaptive_FIR_AN5305.axf"
Erase Done.
Programming Done.
Verify OK.
Flash Load finished at 10:49:38
```

Debugger


File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
WaitForI2C
ADC_GainCompensation
Project
main.c
main.h

RUN

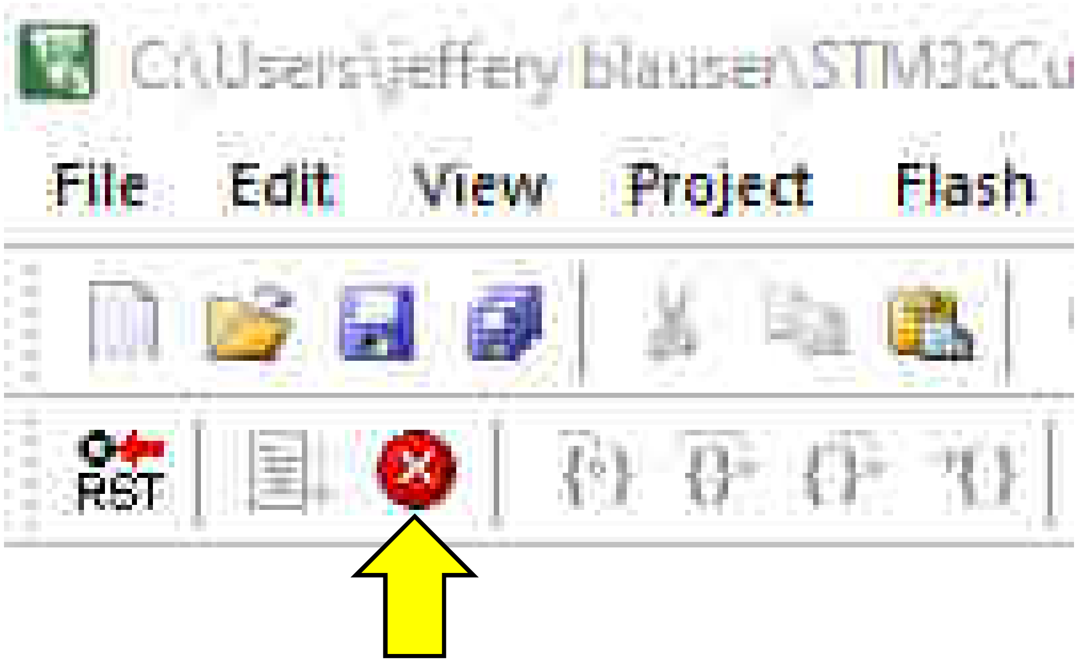




LD4 Illuminates

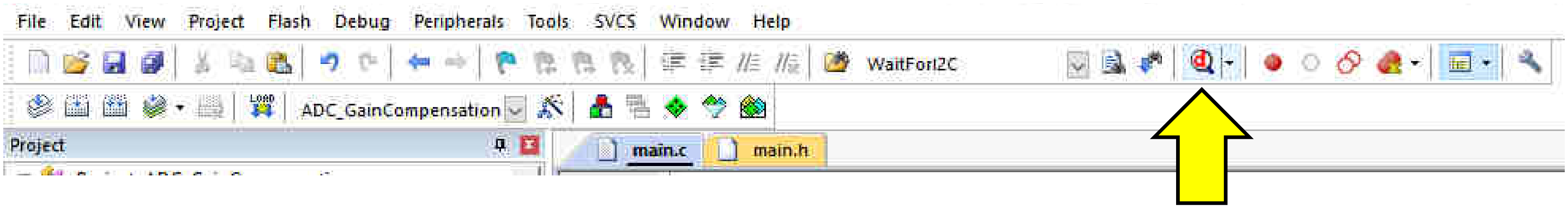
Keil MDK-ARM

STOP Execution



Beyond lighting an indicator LED, let's make it more interesting

Debugger STOP to re-open the Editor



Recall - Lab steps 6 of 6

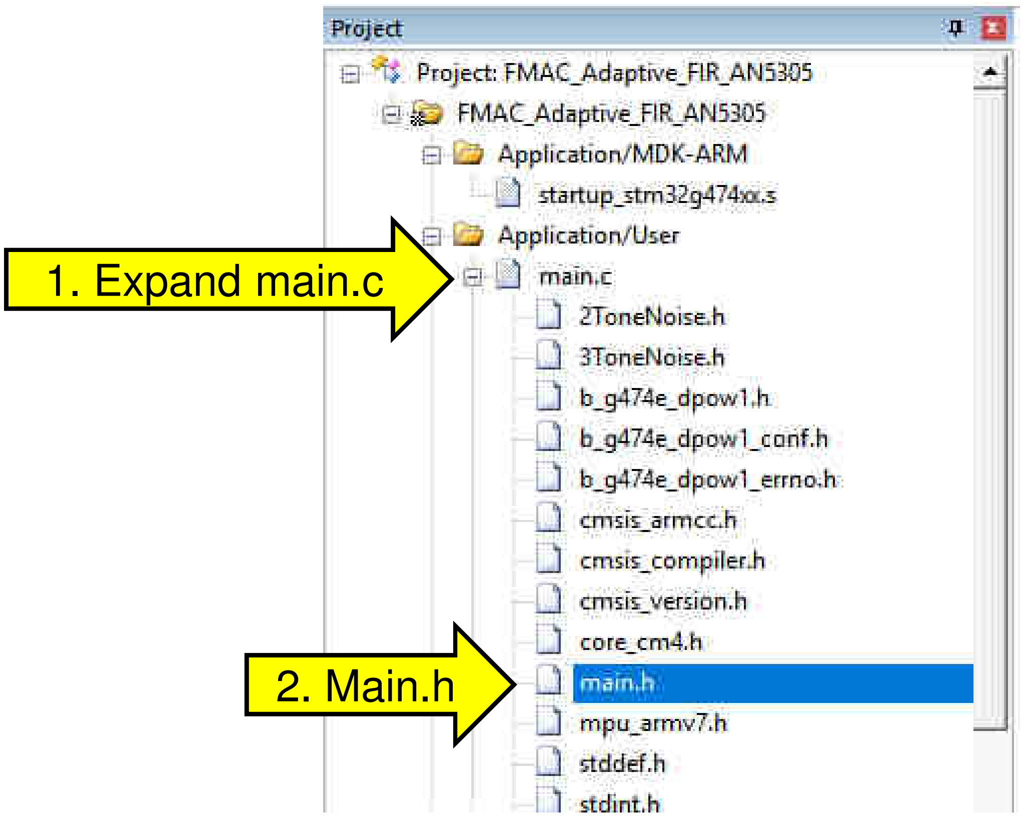
STM32 board LEDs are used to monitor the example status:

- LED4 is ON when correct FMAC FIR results have been calculated.

- LED5 is blinking (1 second period) if an error is detected in the FIR filter results or if there is an initialization or configuration error.

The filtered output is displayed in the Terminal I/O if the compilation switch PRINT_OUTPUT is enabled in main.h.

Finding main.h



Keil MDK-ARM

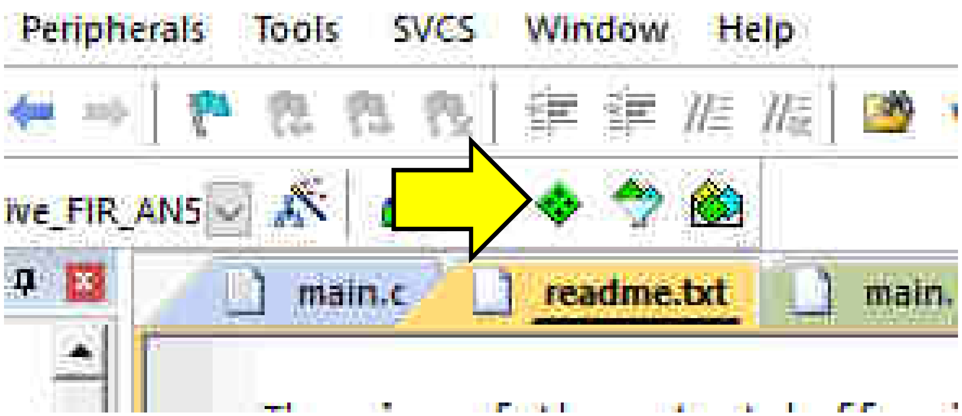
main.h about line 119

```
112 /* FMAC internal memory configuration: base address of the output buffer */
113 #define OUTPUT_BUFFER_BASE       COEFFICIENT_BUFFER_SIZE + INPUT_BUFFER_SIZE
114
115 /* FMAC internal memory configuration: size of the input buffer */
116 #define OUTPUT_BUFFER_SIZE       MEMORY_PARAMETER_D2
117
118 /* Print results to standard IO */
119 #define PRINT_OUTPUT
120
121 /* USER CODE END Private defines */
122
```

Confirm **#define PRINT_OUTPUT** is not commented out

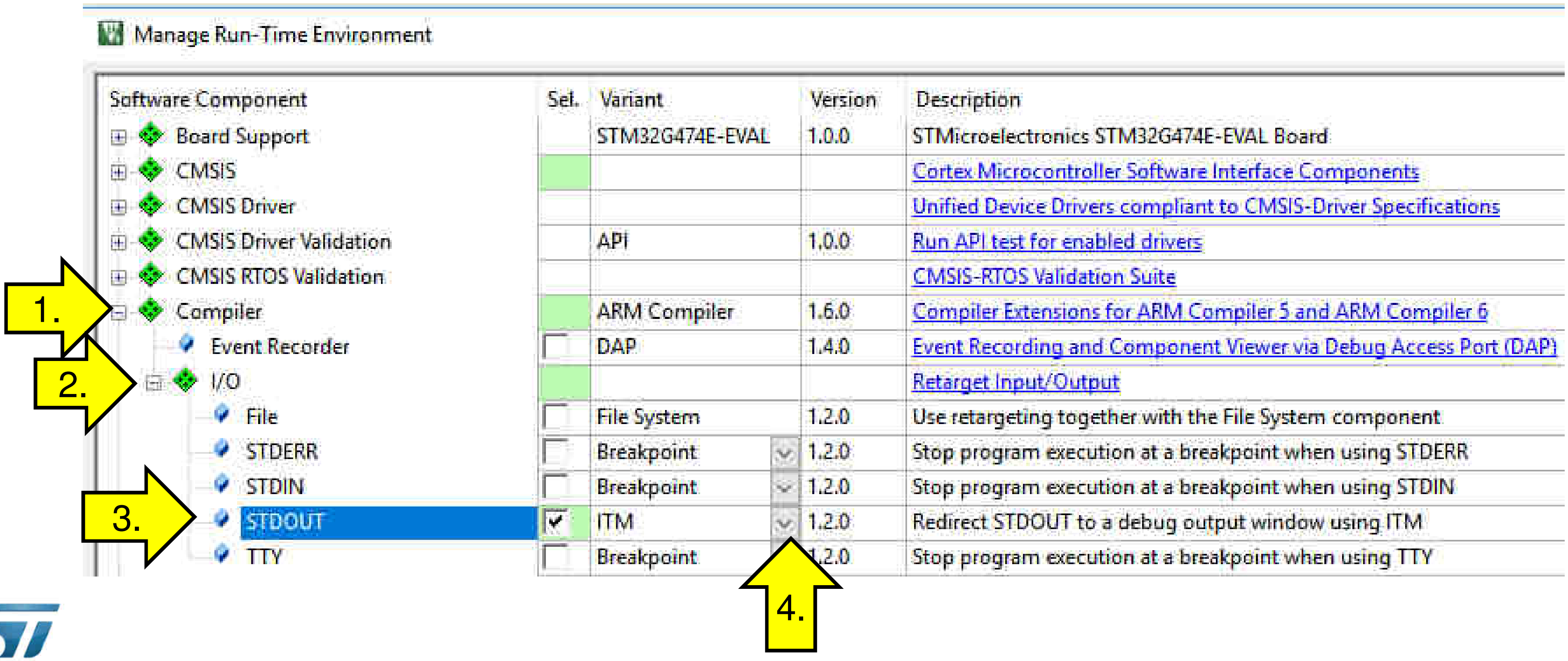
Keil MDK-ARM

Check the Run Time Environment (green diamond)



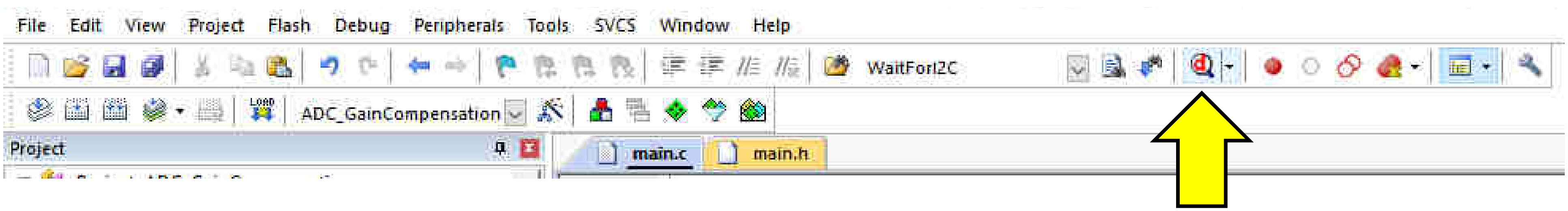
Is stdio enabled to support printf?

Complier – I/O – STDOUT – Check box - ITM

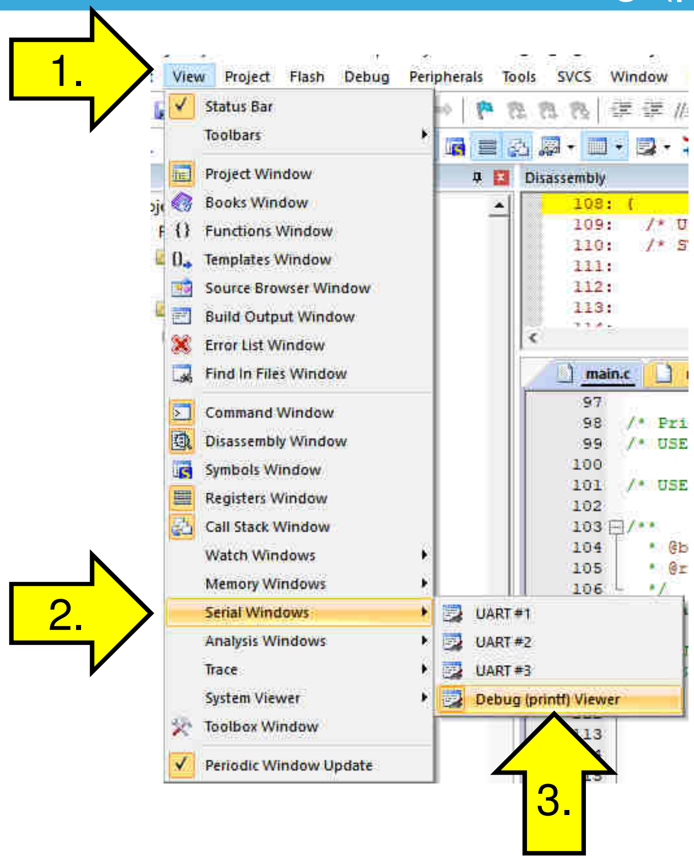


Keil MDK-ARM

Debugger

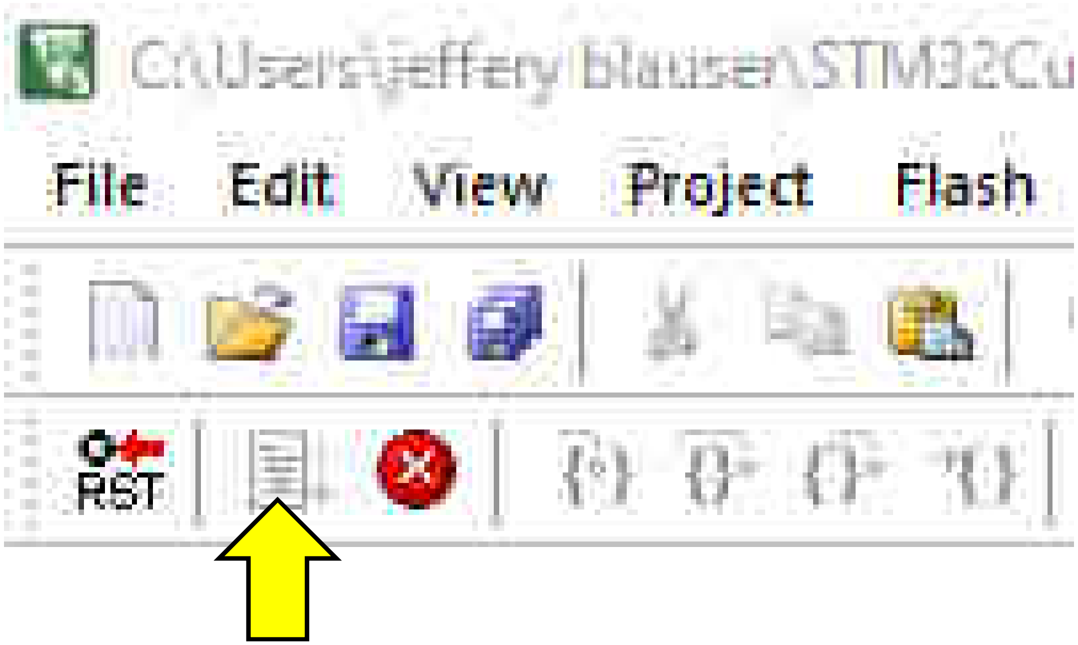


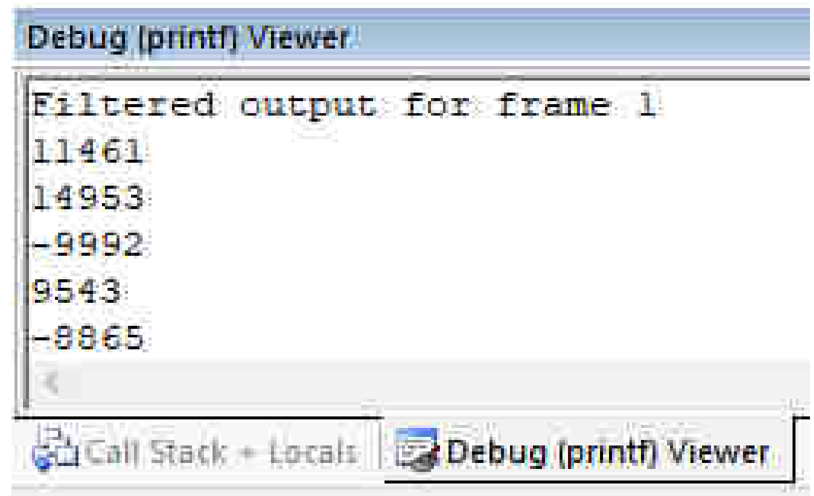
View – Serial Windows – Debug (printf) Viewer



Keil MDK-ARM

RUN – Values in the Debug (printf) Viewer





Filter Math Accelerator

Recal - Lab steps 1 of 6

This Hands On Lab illustrates noise cancellation using an adaptive FIR filter.

The experiment will be run on two input frames:

- a gaussian noise with two tone interferers.
- a gaussian noise with three tone interferers.

Create individual coefficient sets.

The result is a "whitened" noise shown here.

Figure 3. 'Whitened' noise (frame 1)

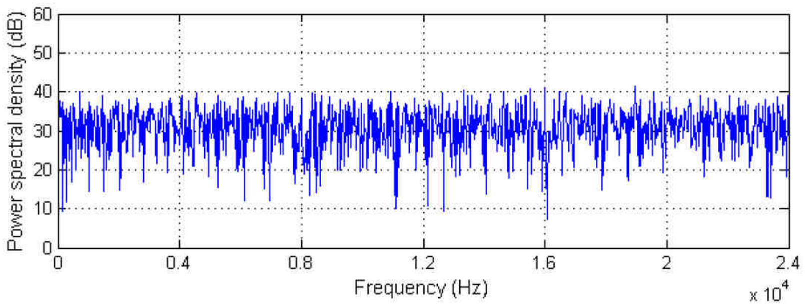
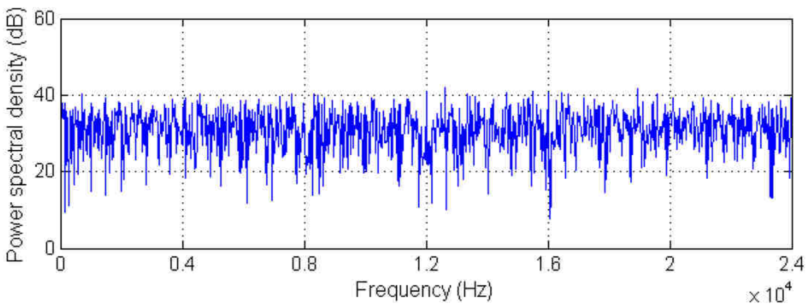
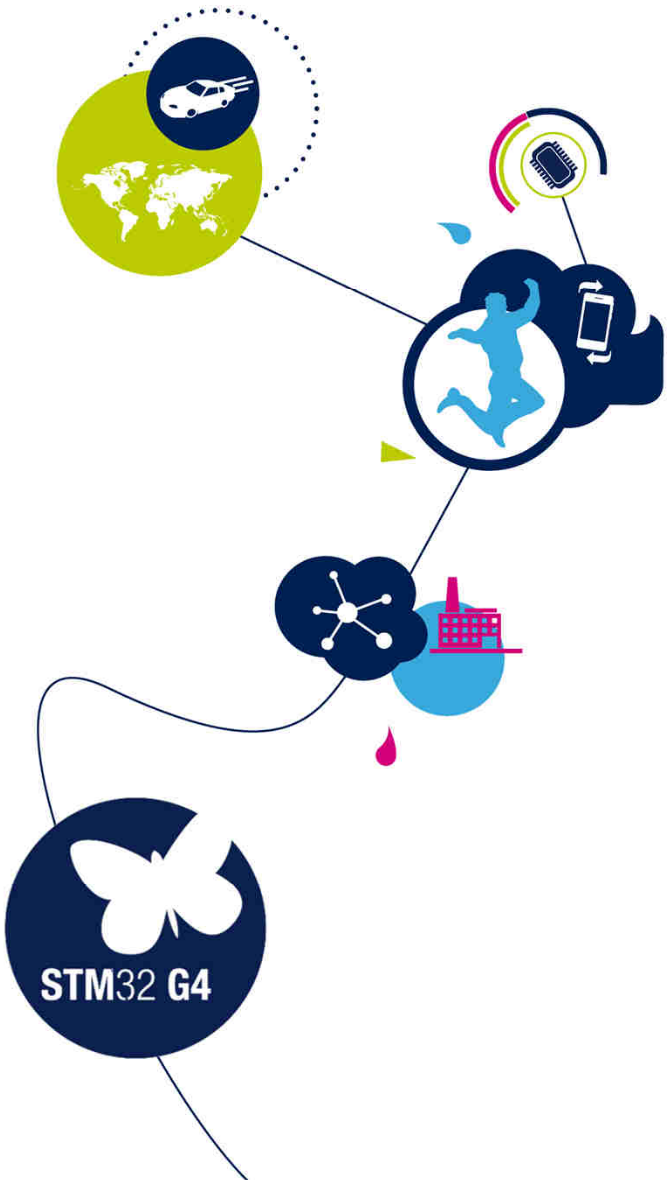


Figure 6. 'Whitened' noise (frame 2)



STM32G4 mixed-signal MCU workshop

Hands On:
Dual Bank Flash
Live Firmware Upgrade



life.augmented

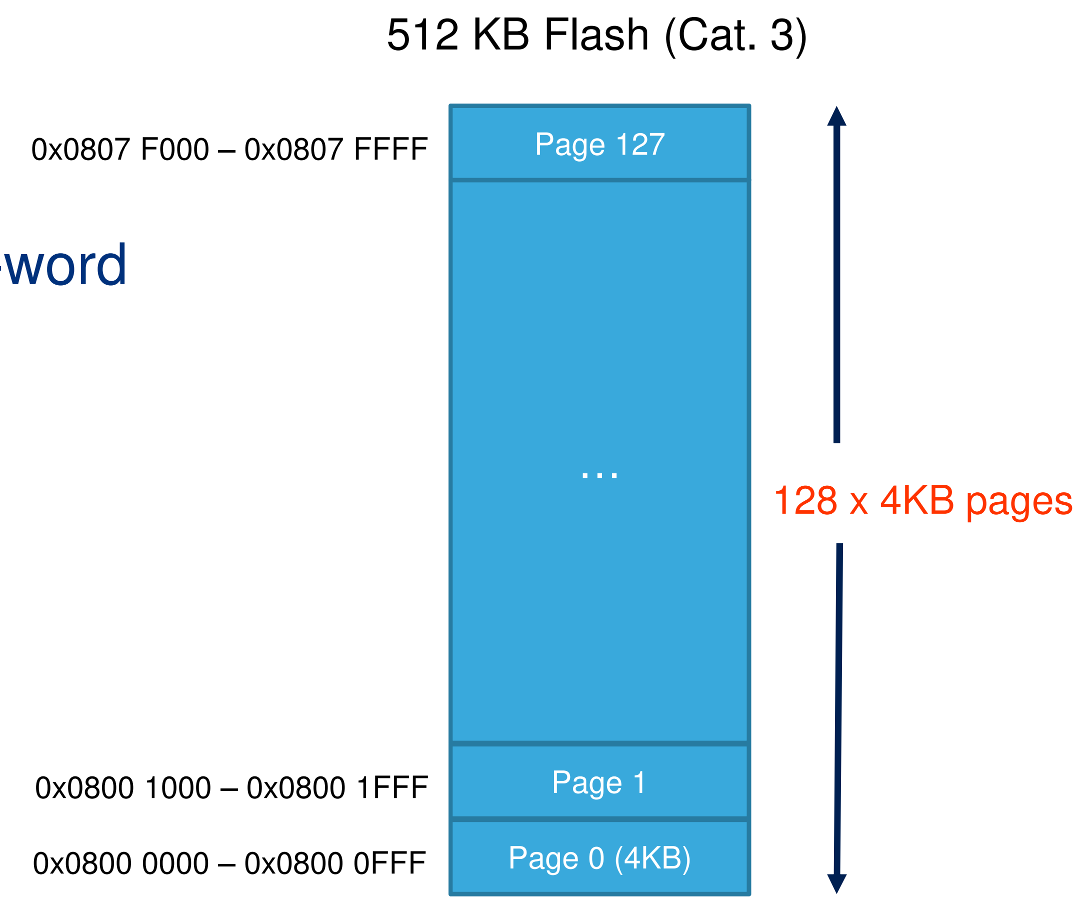
Dual Bank flash

Objective:

- To show the application running from one bank and writing to the second bank(Read-While-Write). Demonstrate the live upgrade possibility.
- Using a Terminal (TeraTerm) to send updated application binary file via the Virtual Comm Port.

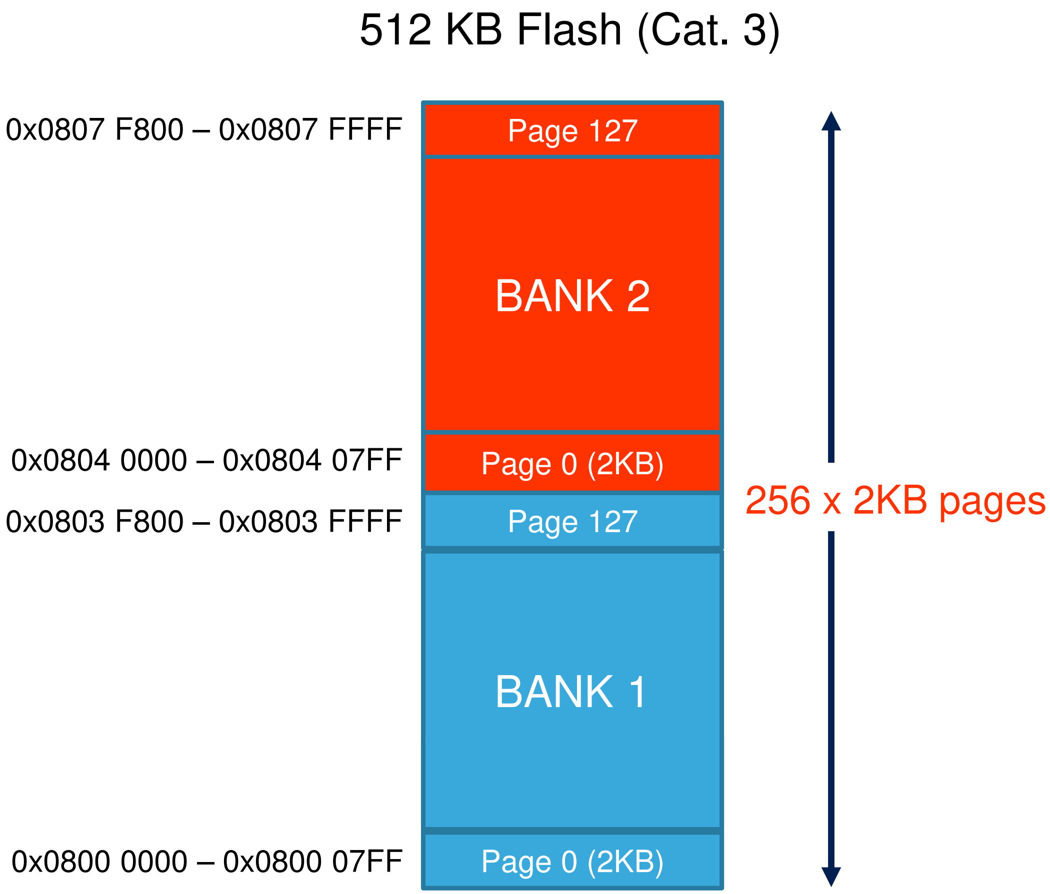
512 KB die Flash Organization

- DBANK = 0 (Single Bank)
  - 128 bits read access
- ECC: 8 bits per 64 bit double-word
- 4 Write protection areas
- Page and Mass erase



512 KB die Flash Organization

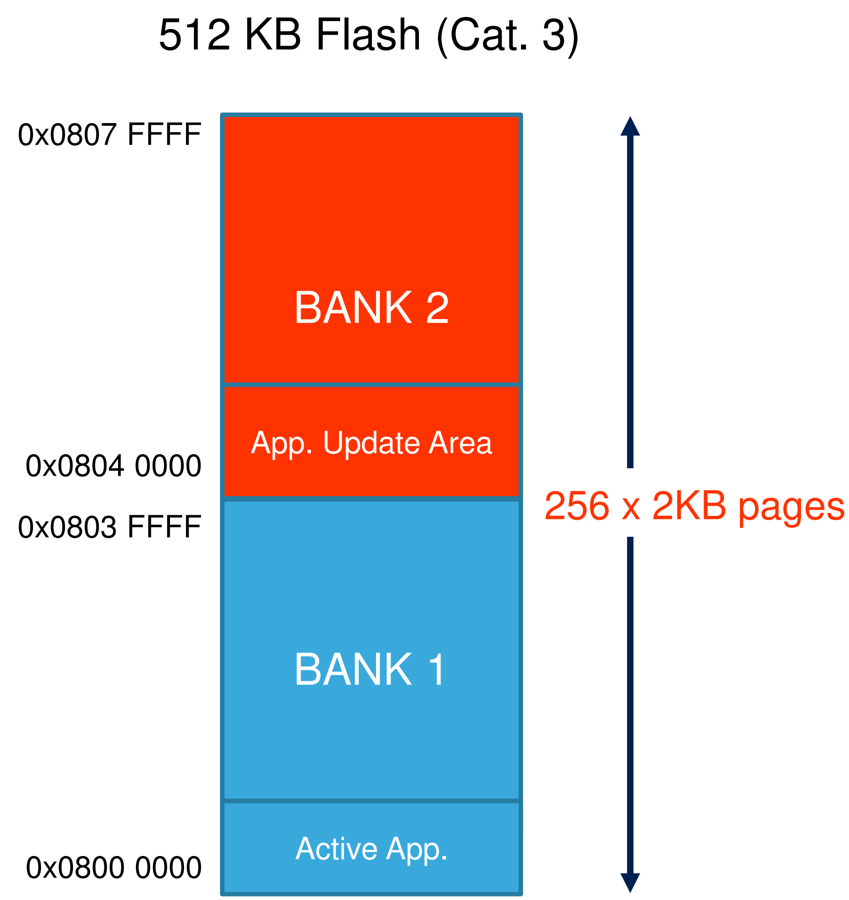
- DBANK = 1 (Dual Bank - Default)
  - 64 bits read access
- Read-While-Write (RWW)
- ECC: 8 bits per 64 bit double-word
- 2 Write protection areas per bank
- Page, Bank and Mass erase



life.augmented

512 KB die Flash Organization

- Active application executing from Bank 1
- Using serial port to receive update
  - Temporary storage in RAM
- Write to BANK 2 area
- Switch application execution to BANK 2

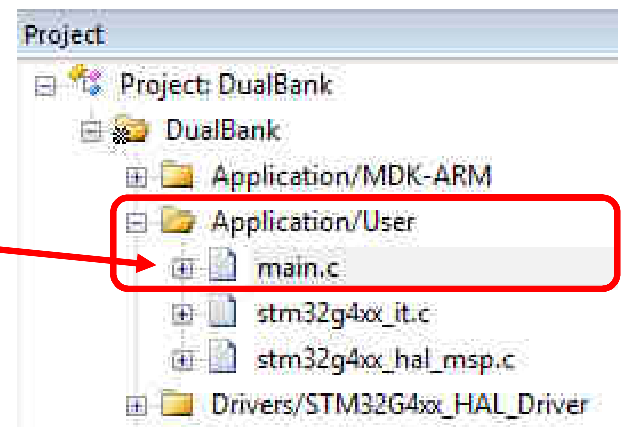


Hands-On: Dual Bank flash

- Open the DualBank project in Keil MDK-ARM folder
  - DualBank\MDK-ARM\DualBank.uvprojx

- Expand "Application/User" folder

- Open "main.c" file
  - Initialize the Clocks, GPIO and Peripherals
  - Main while loop (State Machine)
    - Handles BANK 2 page erase
    - Handles BANK 2 page write
    - Jump and Execute in Bank 2



IRQ Callback functions

- Timer 2 IRQ Callback(Line 481)
  - 10 Hz period
  - LD_ptr: Tracks LED to toggle

```
481   void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
482   {
483     switch ( LD_ptr )
484     {
```

- GPIO EXTI Callback (Line 509)
  - Detects rising edge of JOYSTICK_SEL button

```
509   void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
510   {
511     if ( ( GPIO_Pin == JOYSTICK_SEL_Pin ) && ( UpdateState == 0 ) )
```

- UART Rx complete Callback (Line 524)
  - Signals that the USART Rx DMA has completed

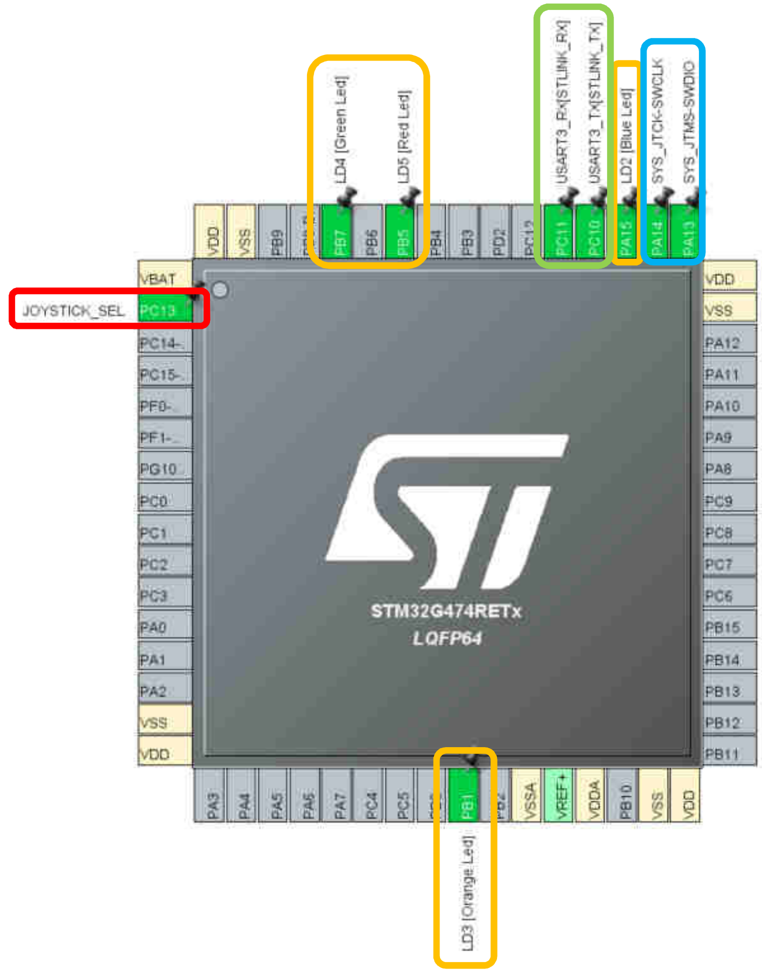
```
524   void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
525   {
526     rx_complete = 1;
527   }
```

STM32 G4

life.augmented

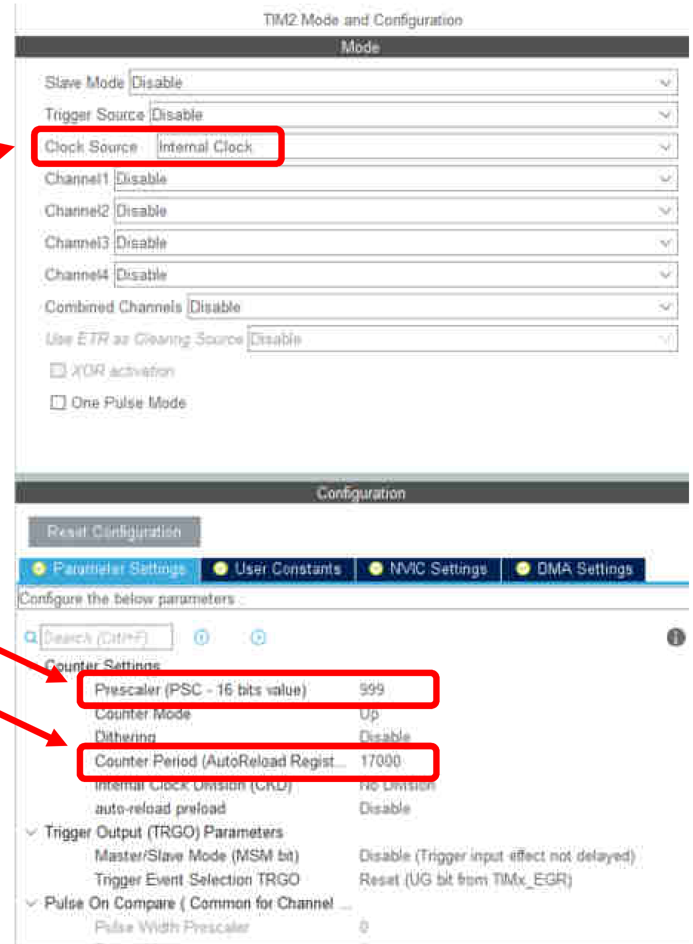
Peripheral configuration

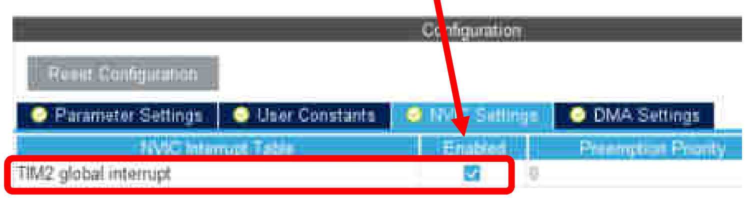
- JOYSTICK_SEL: EXTI13
  - PC13
- LD2-5: GPIO_Output
  - PA15, PB1, PB5 and PB7
- ST-LINK VCP: USART3
  - PC10 and PC13
- Serial Wire Debug with Trace (SWO):
  - PA13, PA14 and PB3



Peripheral configuration

- Timer 2: LED blink rate timer
- Internal Clock Source = 170 MHz
- Prescaler = 999
- Counter Period = 17000
  - 10 Hz timebase interrupt
- TIM2 global interrupt Enabled

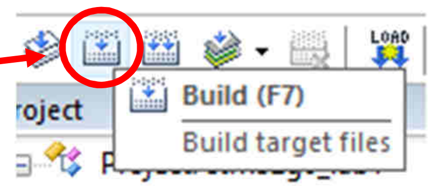




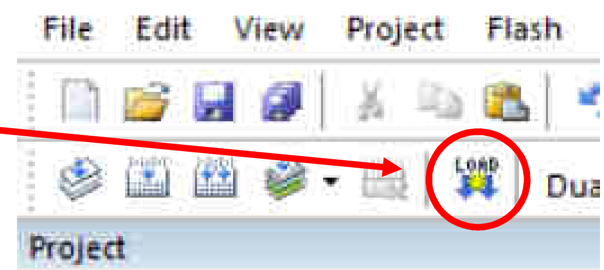
life.augmented

Build the Project

- "Build" project



- "LOAD" project to flash



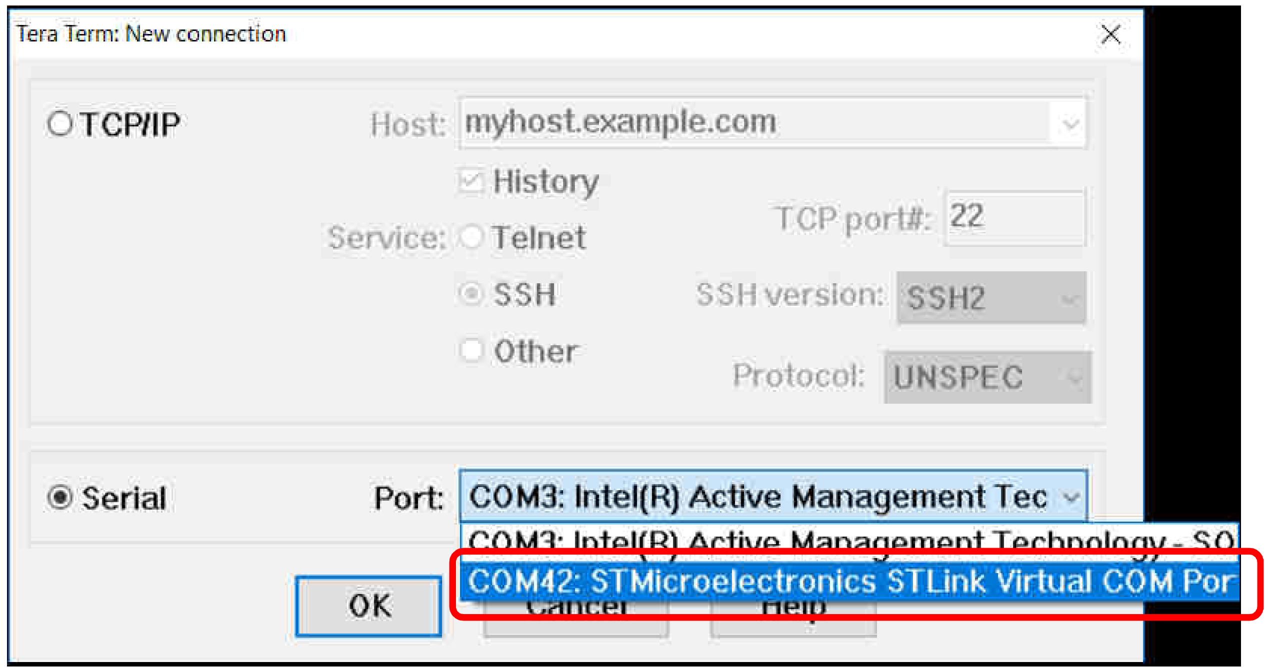
- Press the "RESET" button on the Discovery
  - 4 LEDs blinking in rotating pattern.

life.augmented

Hands On: Dual Bank

- Launch "TeraTerm" 

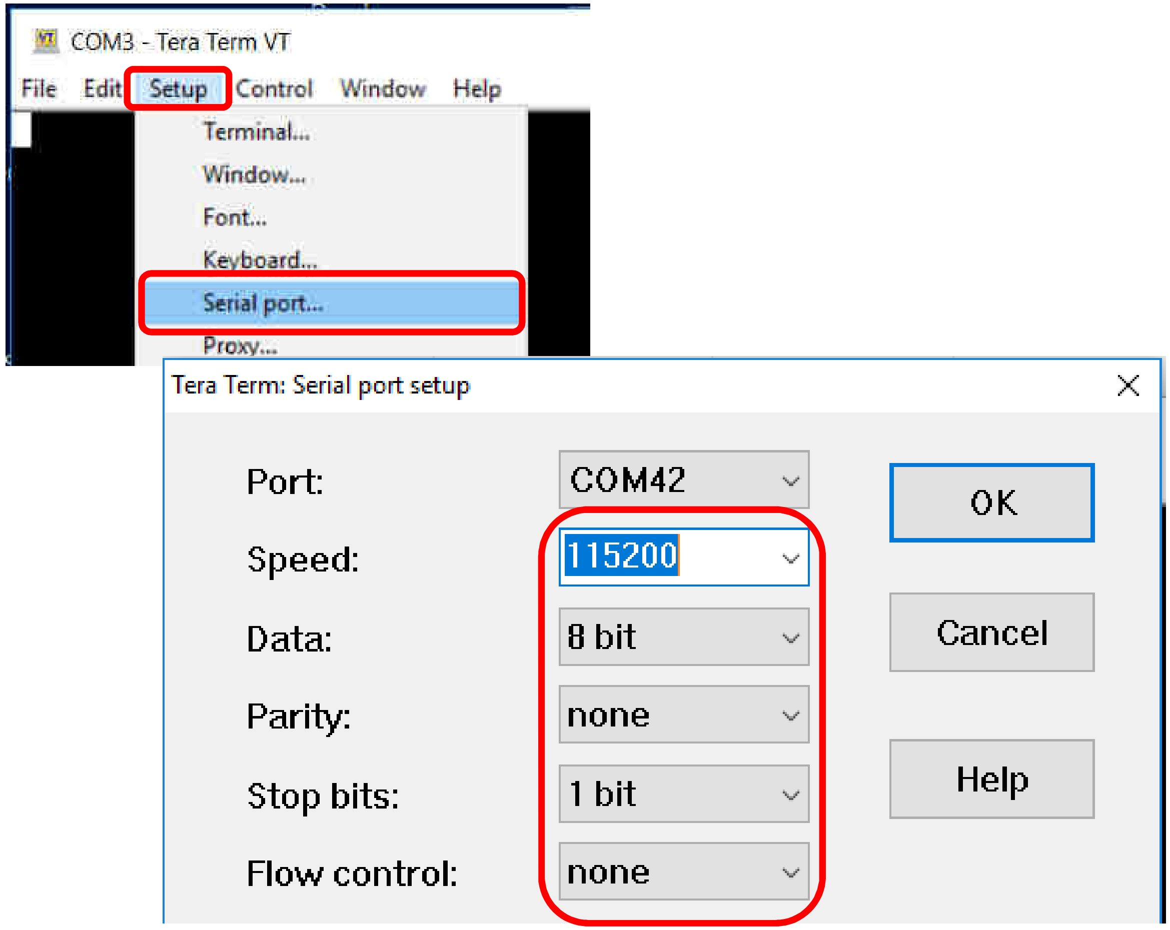
- Open "New Connection"
  - Click "Serial" button
  - Change Port to "STLink Vitual COM Port"
  - Click "OK"



life.augmented

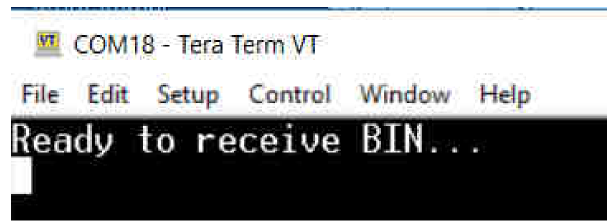
Hands On: Dual Bank

- Verify “Serial port..”
  - Baud = 115200
  - Data = 8 bit
  - Parity = none
  - Stop = 1 bit
  - Flow control = none

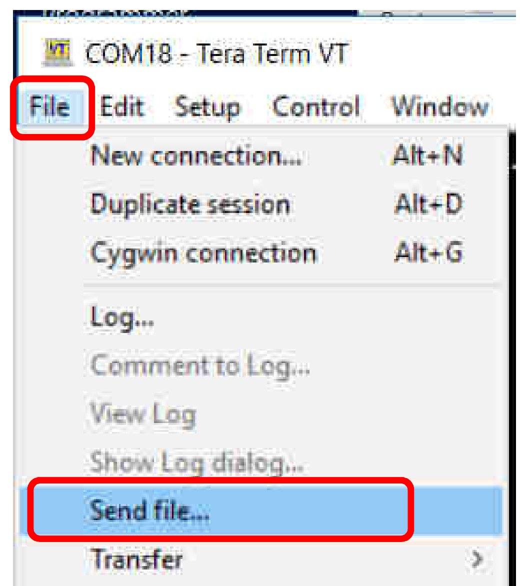


Hands On: Dual Bank

- Press “JOYSTICK_SEL” on Discovery

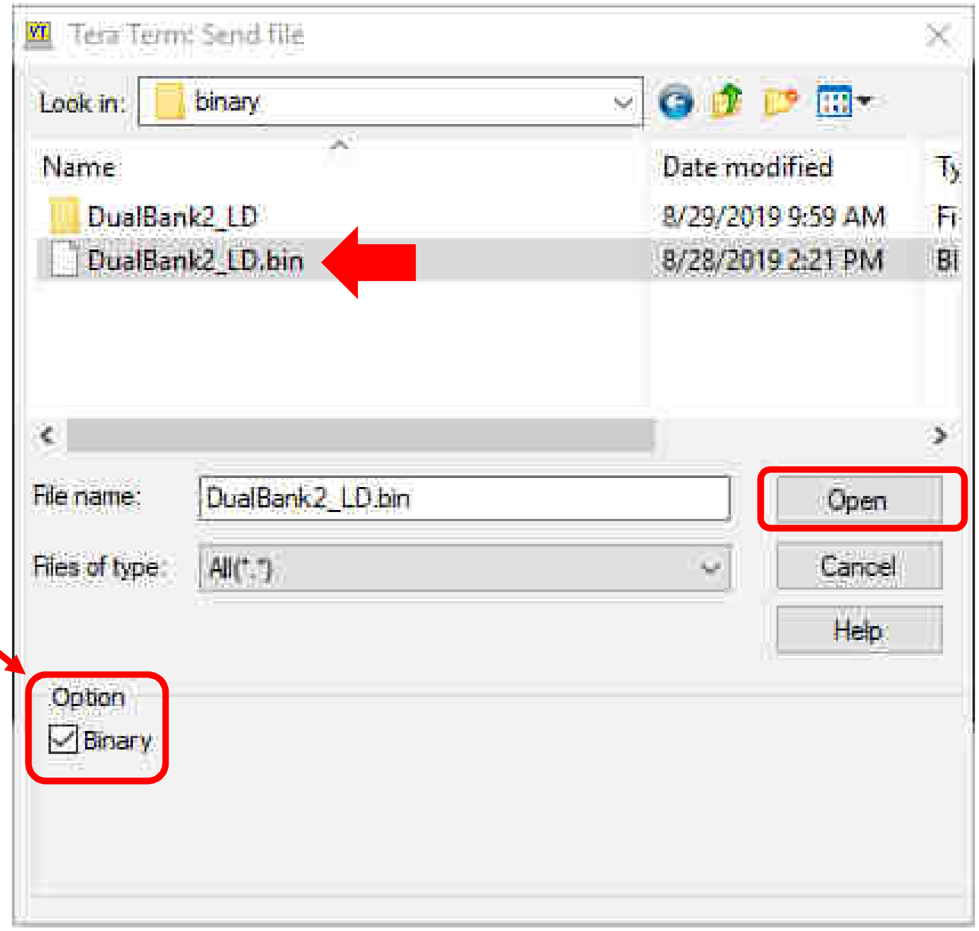


- Send binary file in TeraTerm
  - File > Send file…



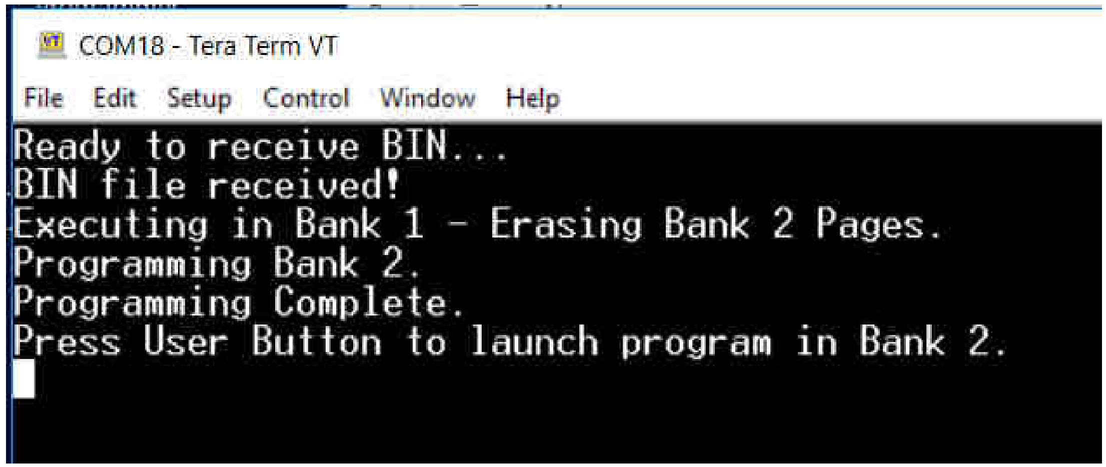
Hands On: Dual Bank

- Select “DualBank2_LD.bin” file
  - DualBank\binary folder
- Check “Binary” Option
- Click “Open”



Hands On: Dual Bank

- TeraTerm will display programming status



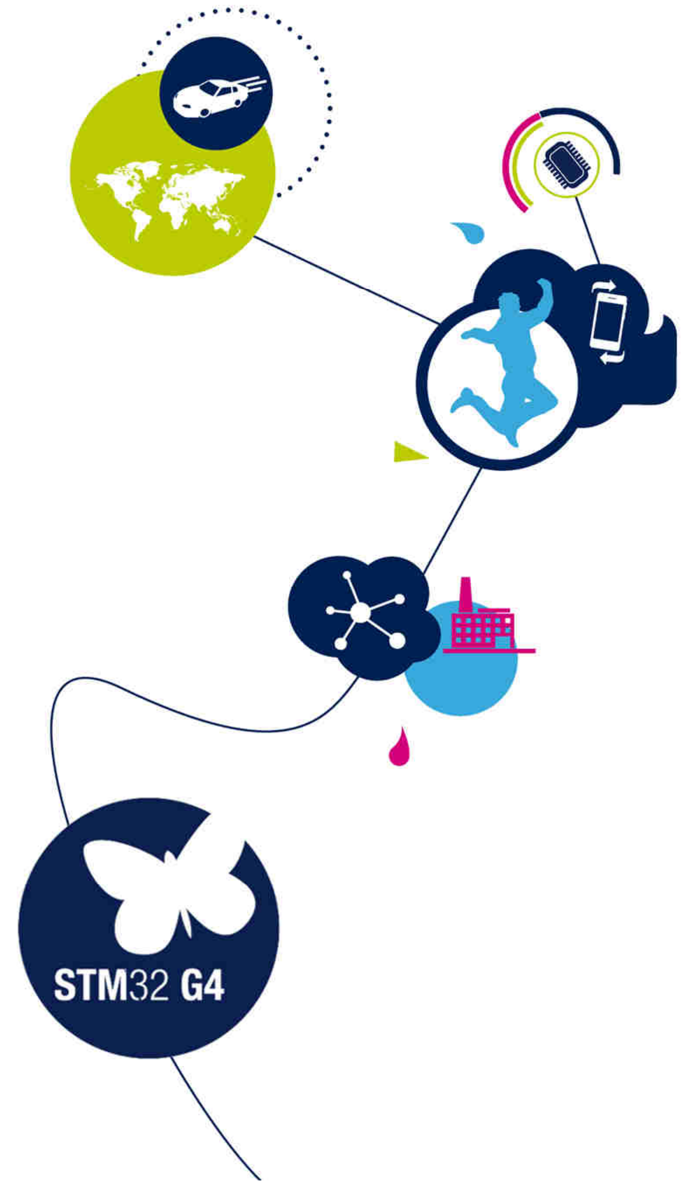
- Press “User Button” to launch program in Bank 2
  - LEDs blink pattern has changed.

Dual Bank: Additional Notes

- Push board “RESET” button to switch execution back to Bank 1
  - To select boot from Bank 2, set the BFB2 bit in the user option bytes
- Reference Manual RM0440: STM32G4 Series advanced ARM-based 32-bit MCUs
  - Section 3: Embedded Flash memory for category 3 devices
- AN2606: STM32 microcontroller system memory boot mode

STM32G4 mixed-signal MCU workshop

Hands On: CCM SRAM



life.augmented

Hands-On: CCM SRAM

Objective:

1. Illustrate how to place functions in CCM SRAM.
2. Illustrate how to place files in CCM SRAM.
3. Illustrate how to place library modules in CCM SRAM.

CCM SRAM Advantages

- Tightly coupled to ARM Cortex core
- Zero wait state with deterministic performance
- Decrease execution time for critical task
- Minimal Interrupt routines latency when placed in CCM SRAM

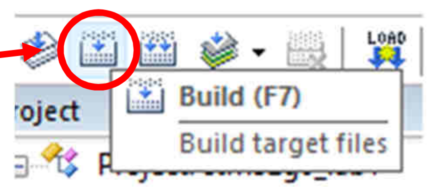
CCM SRAM (functions)

- Hands-On: Place function in CCM SRAM
- Open the CCM1.ioc CubeMx project (double mouse click)
  - Labs\CCM\CCM1\CCM1.ioc

- Generate Code

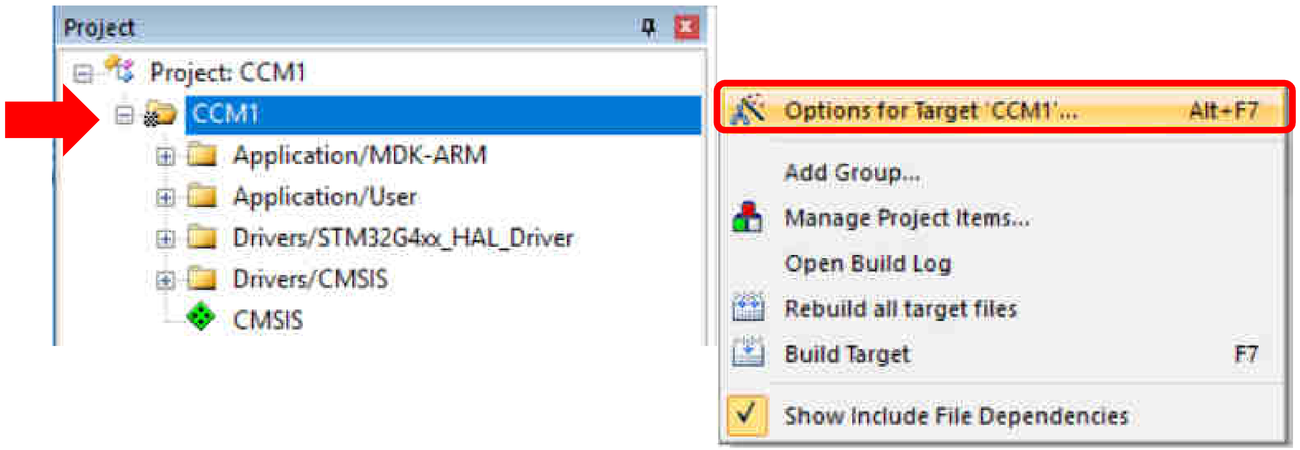
- Open Project

- Build Project



CCM SRAM (functions)

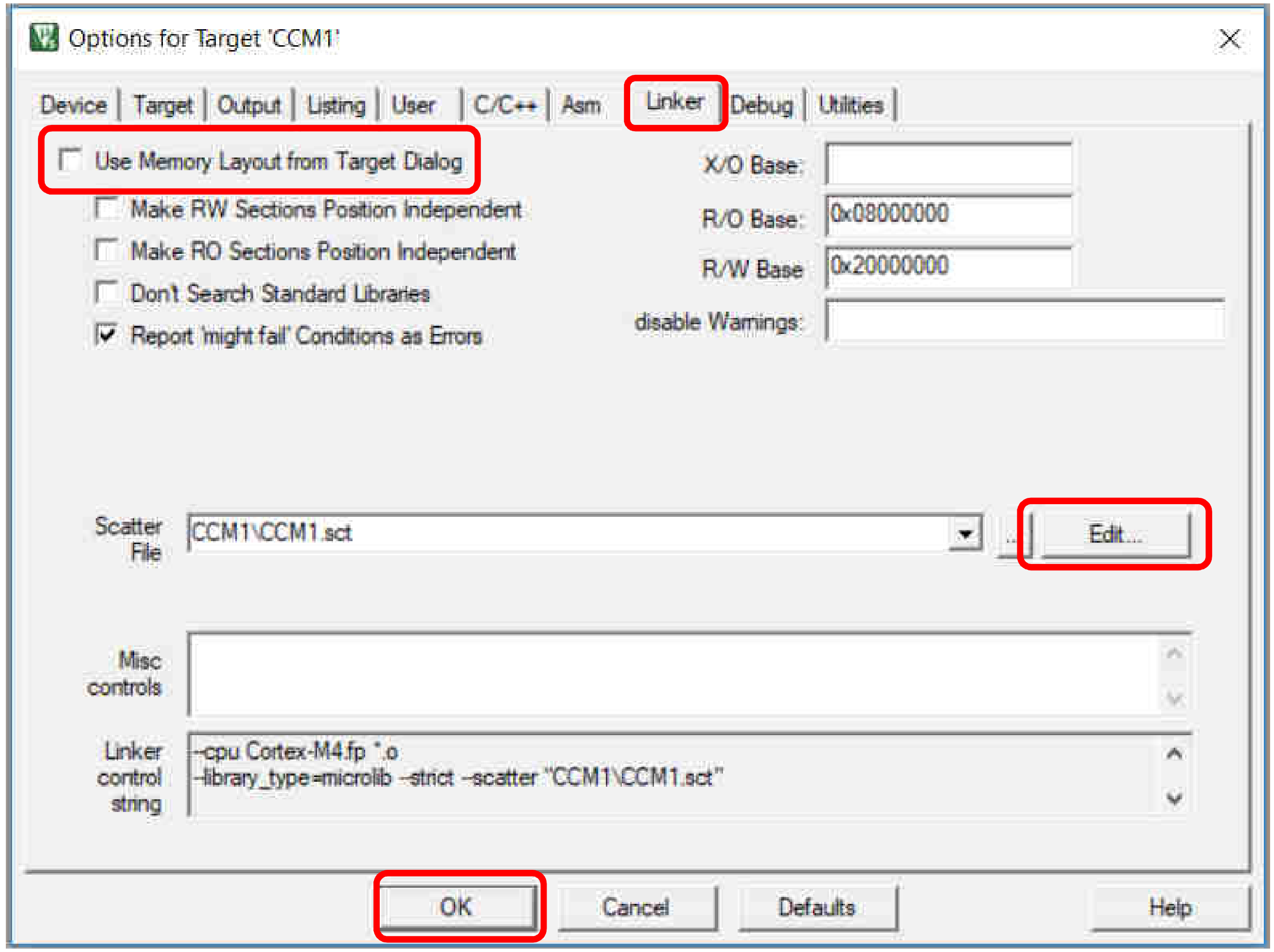
- Define a new memory region for the CCM SRAM in the MDK-ARM project scatter file
  - Right mouse click on CCM1
  - Select “Options for Taget ‘CCM1’…



CCM SRAM (functions)

STM32 G4

- Select “Linker” tab
- Uncheck “Use Memory Layout from Target Dialog”
- Select “Edit” to open scatter file
- Click “OK”



life.augmented

CCM SRAM (functions)

- Place a new line after line 14 (RW_IRAM1 region)
- Define a new CCMRAM region with the following:

```
ER 0x10000000 0x00002000    {
  .ANY (ccmram)
}
```

- Save CCM1.sct file

CCM1.sct

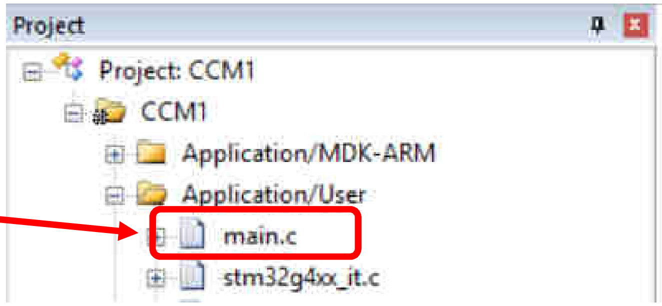
```
1   ; *************************************************************
2   ; *** Scatter-Loading Description File generated by uVision ***
3   ; *************************************************************
4
5   LR_IROM1 0x08000000 0x00080000  {    ; load region size_region
6     ER_IROM1 0x08000000 0x00080000  {  ; load address = execution address
7      *.o (RESET, +First)
8      *(InRoot$$Sections)
9      .ANY (+RO)
10     .ANY (+XO)
11    }
12    RW_IRAM1 0x20000000 0x00020000  {  ; RW data
13     .ANY (+RW +ZI)
14    }
15    ER 0x10000000 0x00002000    {    ; load address = execution address
16     .ANY (ccmram)
17    }
18  }
19
```

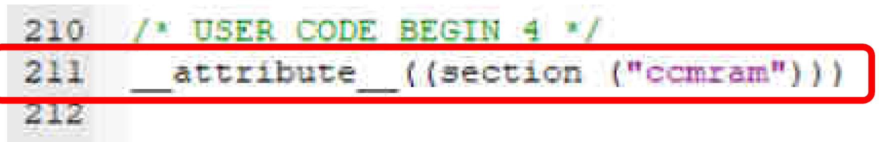
CCM SRAM (functions)

- Expand “Application/User” folder
- Open “main.c” file
  - Scroll down to line 211
- Place the MDK-ARM keyword

__attribute__ ((section (“ccmram”)))

- Place keyword above each function to execute from CCM SRAM





CCM SRAM (functions)

- Place function to execute from CCM SRAM after __attribute__ keyword
  - Copy and paste “HAL_GPIO_EXTI_Callback” function from CCM lab code.txt to the start of line 214

```
210 /* USER CODE BEGIN 4 */
211 __attribute__((section ("ccmram")))
212
213
214 void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
215 {
216   if ( GPIO_Pin == JOYSTICK_SEL_Pin )
217   {
218     HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, GPIO_PIN_RESET);
219     HAL_GPIO_WritePin(LD3_GPIO_Port, LD3_Pin, GPIO_PIN_RESET);
```

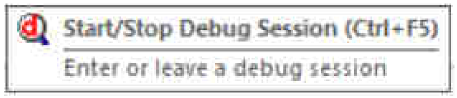
STM32 G4

life.augmented

CCM SRAM (functions)

- "Build" project

- "Start Debug" Session



- Place breakpoint in "HAL_GPIO_EXTI_Callback" (Line 216)

```
214   void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
215   {
216     if ( GPIO_Pin == JOYSTICK_SEL_Pin )
217     {
```

- "Run" code

- Press Joystick button (Up, Down, Left, Right, Select)
- Breakpoint stops in CCM SRAM region address 0x10000018

```
222:     }
0x10000018 F5B05F00   CMP             r0,#0x2000
215: {
```

CCM SRAM (files)

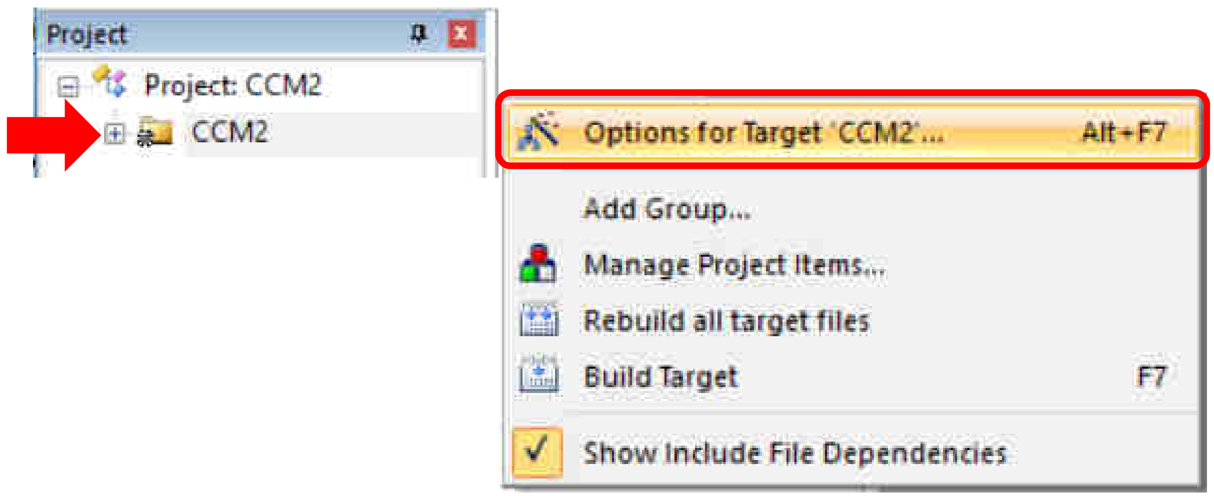
- Hands-On: Place file(s) in CCM SRAM
- Open the CCM2.ioc CubeMx project (double mouse click)
  - Labs\CCM\CCM2\CCM2.ioc

- Generate Code

- Open Project

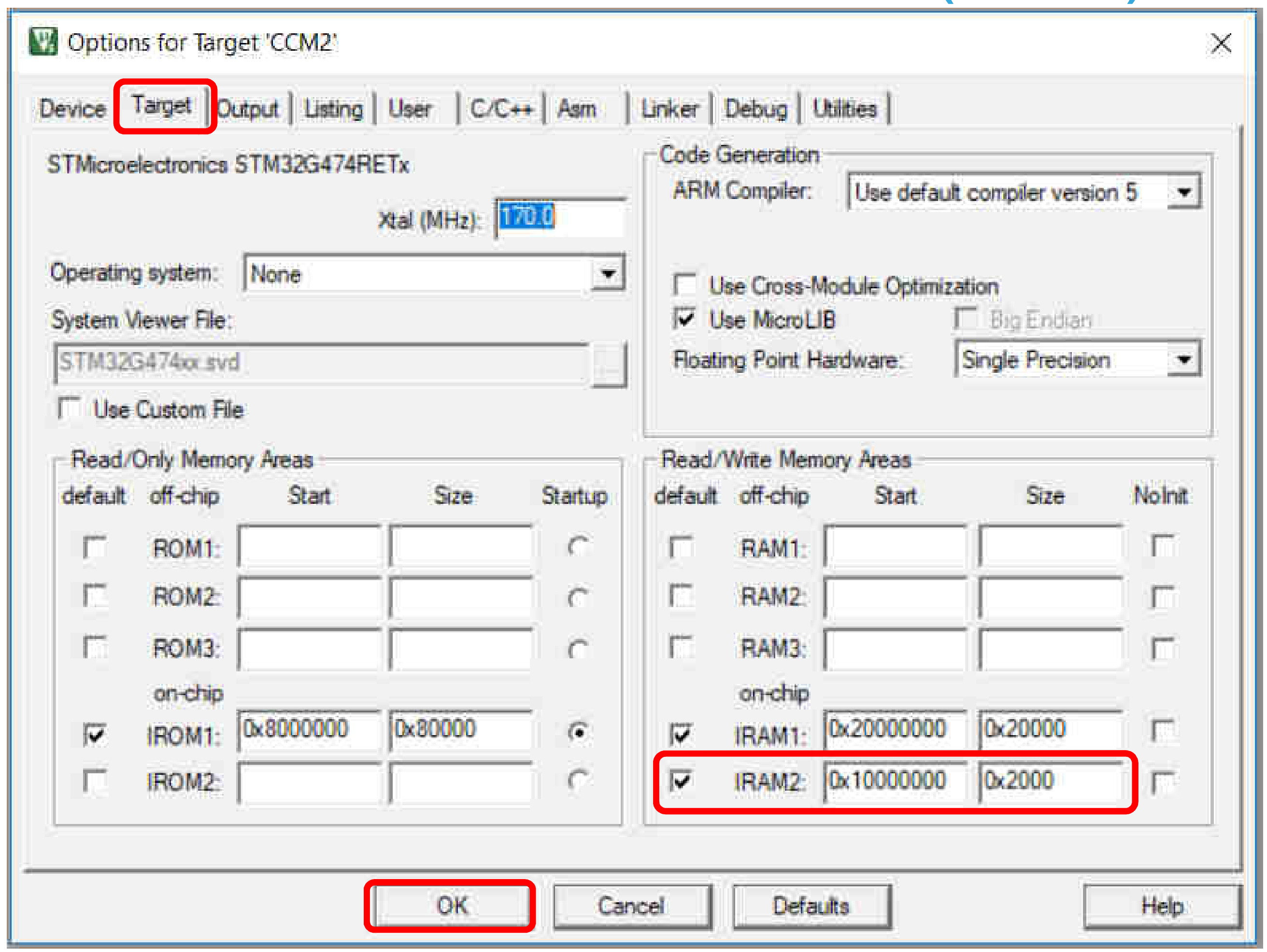
CCM SRAM (files)

- Define the memory region for the CCM SRAM in the MDK-ARM
  - Right mouse click on CCM2
  - Select “Options for Taget ‘CCM2’…



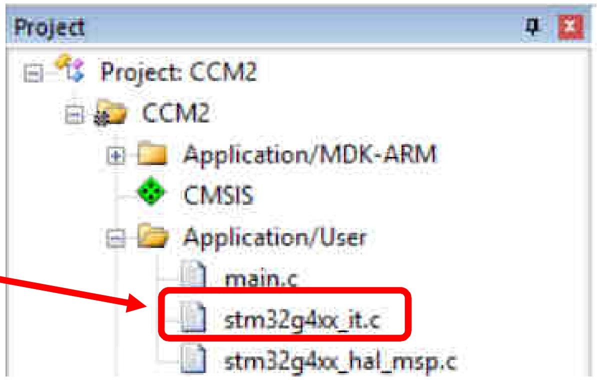
CCM SRAM (files)

- Select “Target” tab
- Check “IRAM2”
- Enter “0x10000000” IRAM2 Start
- Enter “0x2000” IRAM2 Size
- Click “OK”



CCM SRAM (files)

- Expand “Application/User” folder
- Open “stm32g4xx_it.c” file
- Copy and paste “HAL_GPIO_EXTI_Callback” function from CCM lab code.txt to the start of line 258

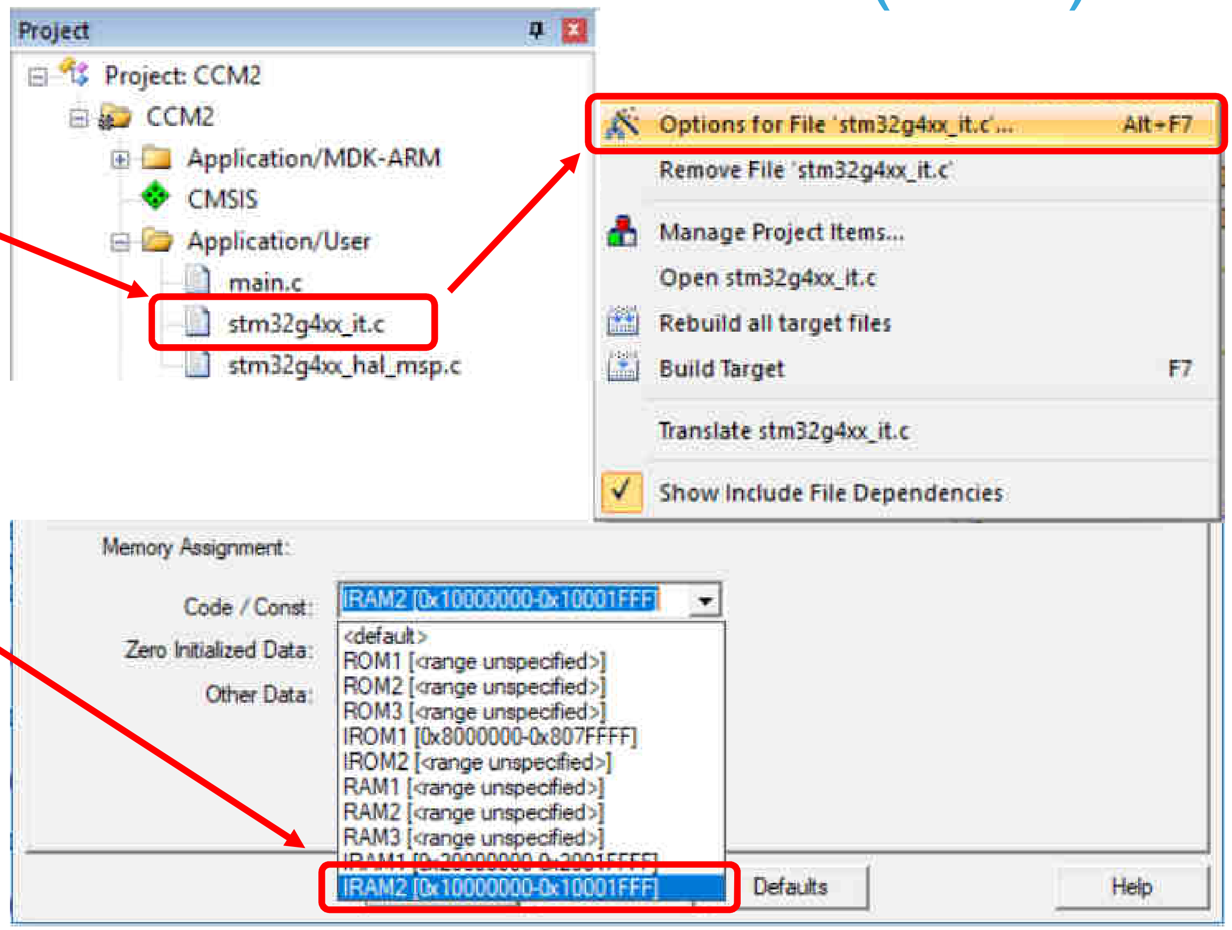



```
257 /* USER CODE BEGIN 1 */
258 void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
259 {
260   if ( GPIO_Pin == JOYSTICK_SEL_Pin )
261   {
262     HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, GPIO_PIN_RESET);
263     HAL_GPIO_WritePin(LD3_GPIO_Port, LD3_Pin, GPIO_PIN_RESET);
```

life.augmented

CCM SRAM (files)

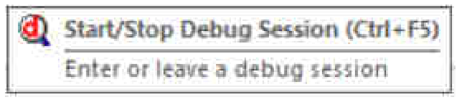
- Right mouse click on “stm32g4xx_it.c”
- Select “Options for File ‘stm32g4xx_it.c’…
- Set Memory Assignment to “IRAM2”
- Click “OK”



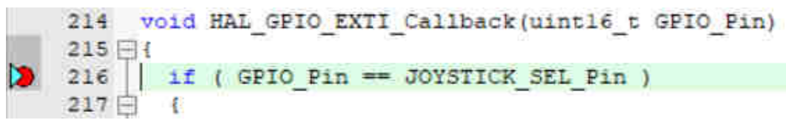
CCM SRAM (files)

- “Build” project

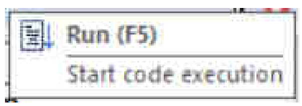
- “Start Debug” Session



- Place breakpoint in “HAL_GPIO_EXTI_Callback” (Line 260)



- “Run” code

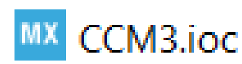


- Press Joystick button (Up, Down, Left, Right, Select)
- Breakpoint stops in CCM SRAM region address 0x10000058

life.augmented

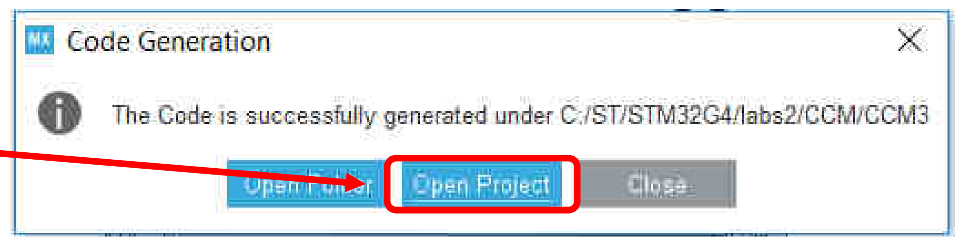
CCM SRAM (library modules)

- Hands-On: Place library module in CCM SRAM
- Open the CCM3.ioc CubeMx project (double mouse click)
  - Labs\CCM\CCM3\CCM3.ioc



- Generate Code

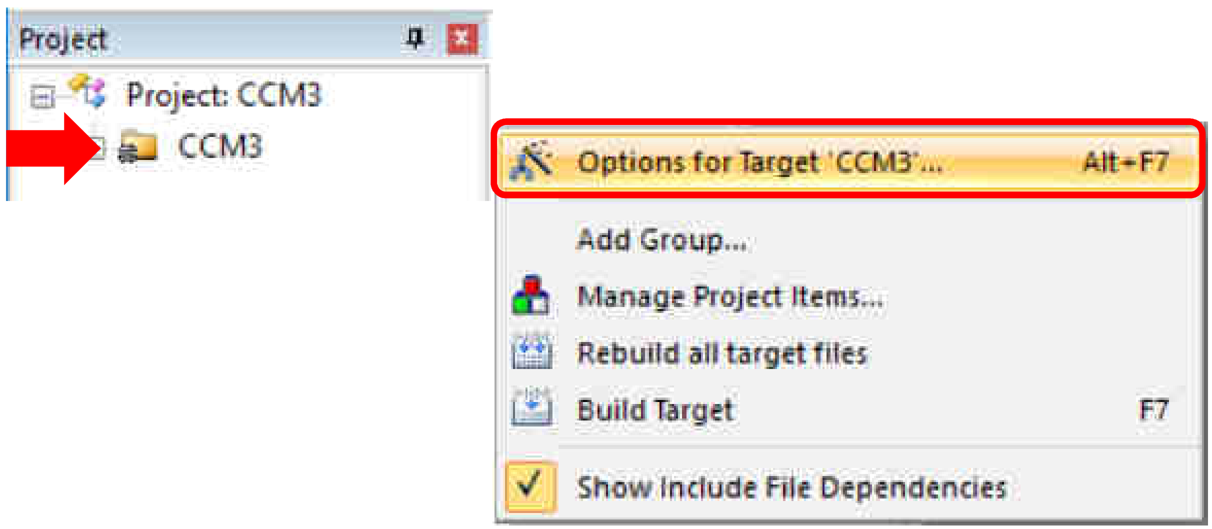
- Open Project



life.augmented

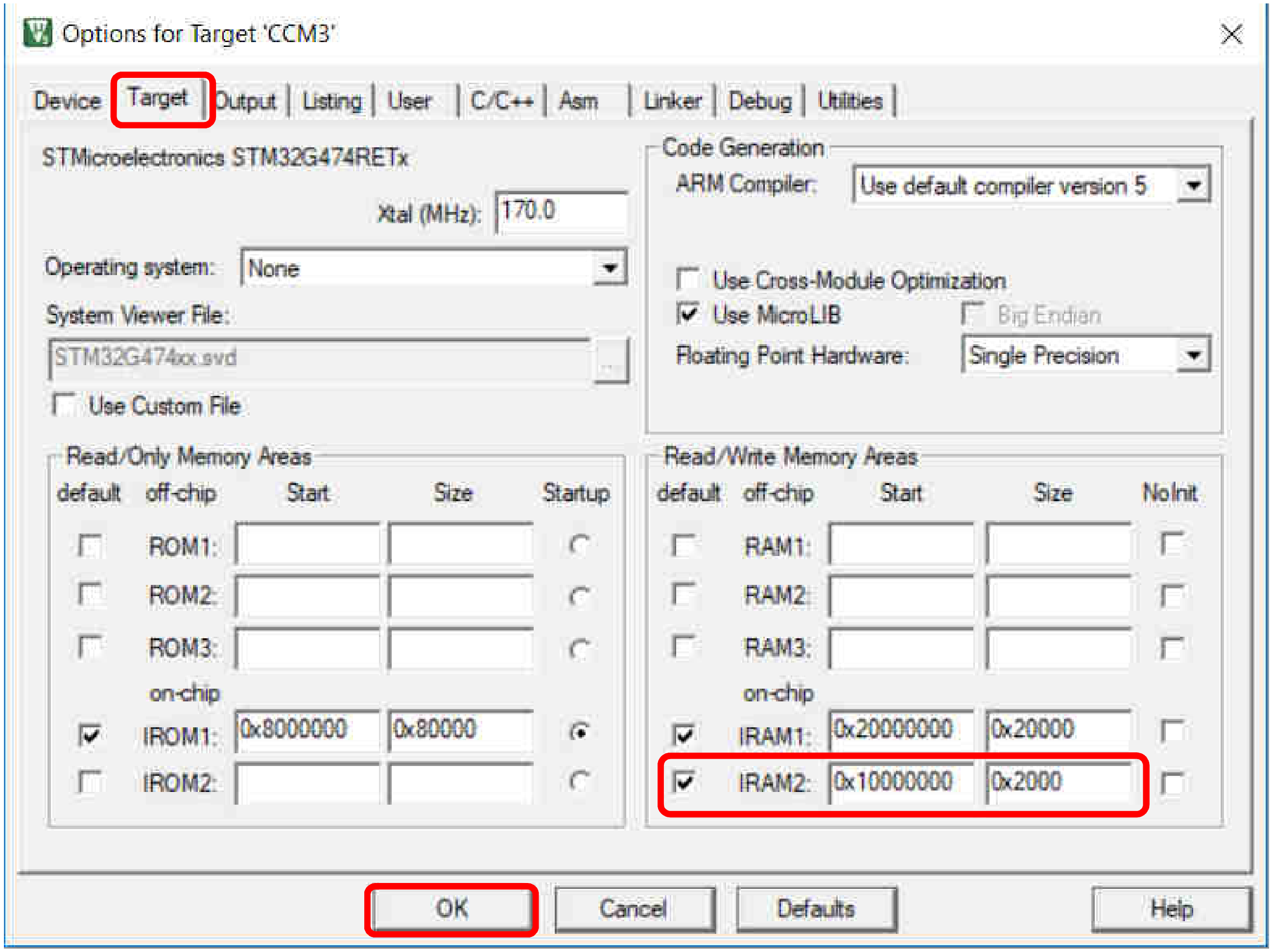
CCM SRAM (library modules)

- Define the memory region for the CCM SRAM in the MDK-ARM
  - Right mouse click on CCM3
  - Select “Options for Target ‘CCM3’…



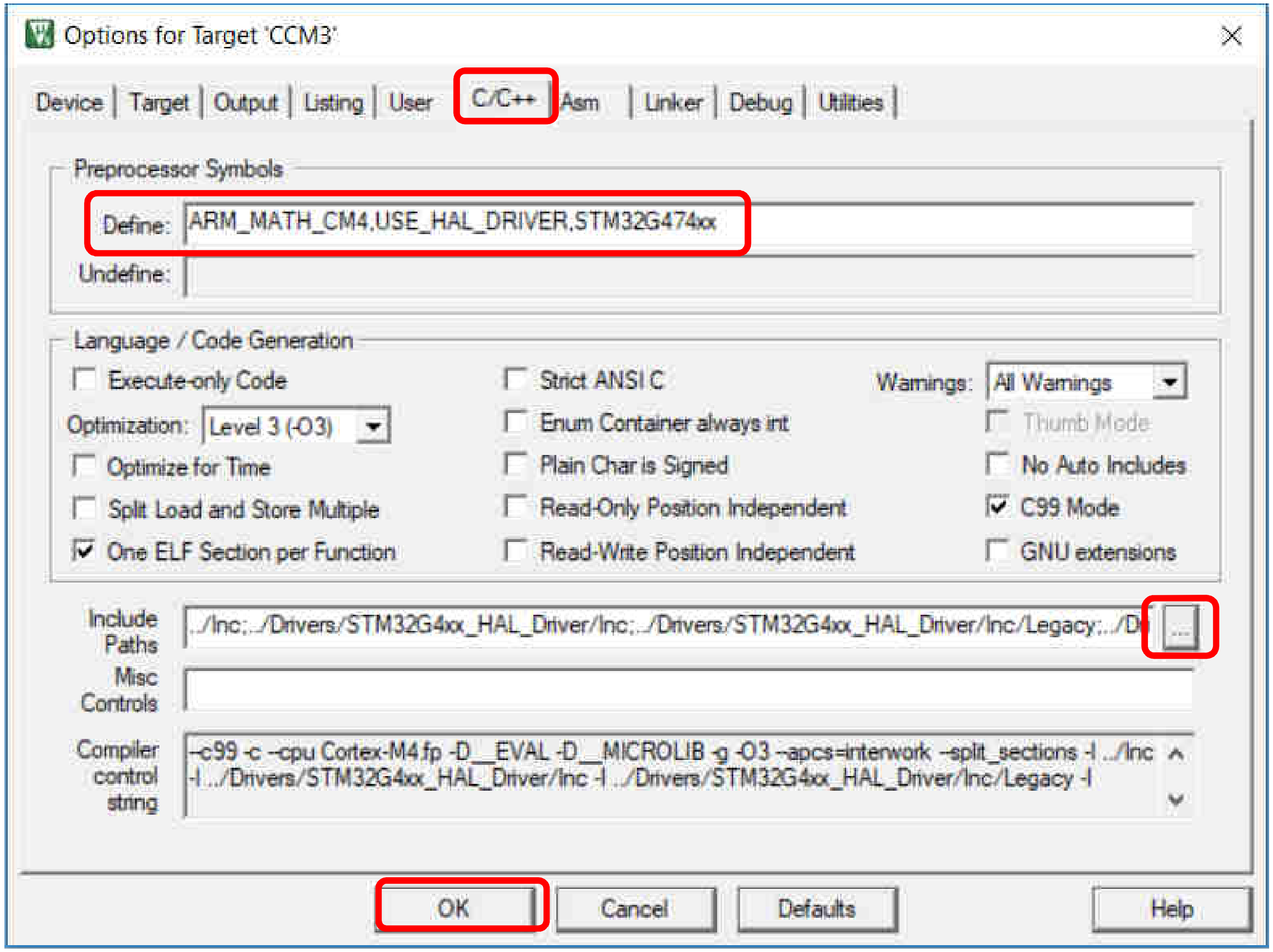
CCM SRAM (library modules)

- Select "Target" tab
- Check "IRAM2"
- Enter "0x10000000" IRAM2 Start
- Enter "0x2000" IRAM2 Size
- Click "OK"



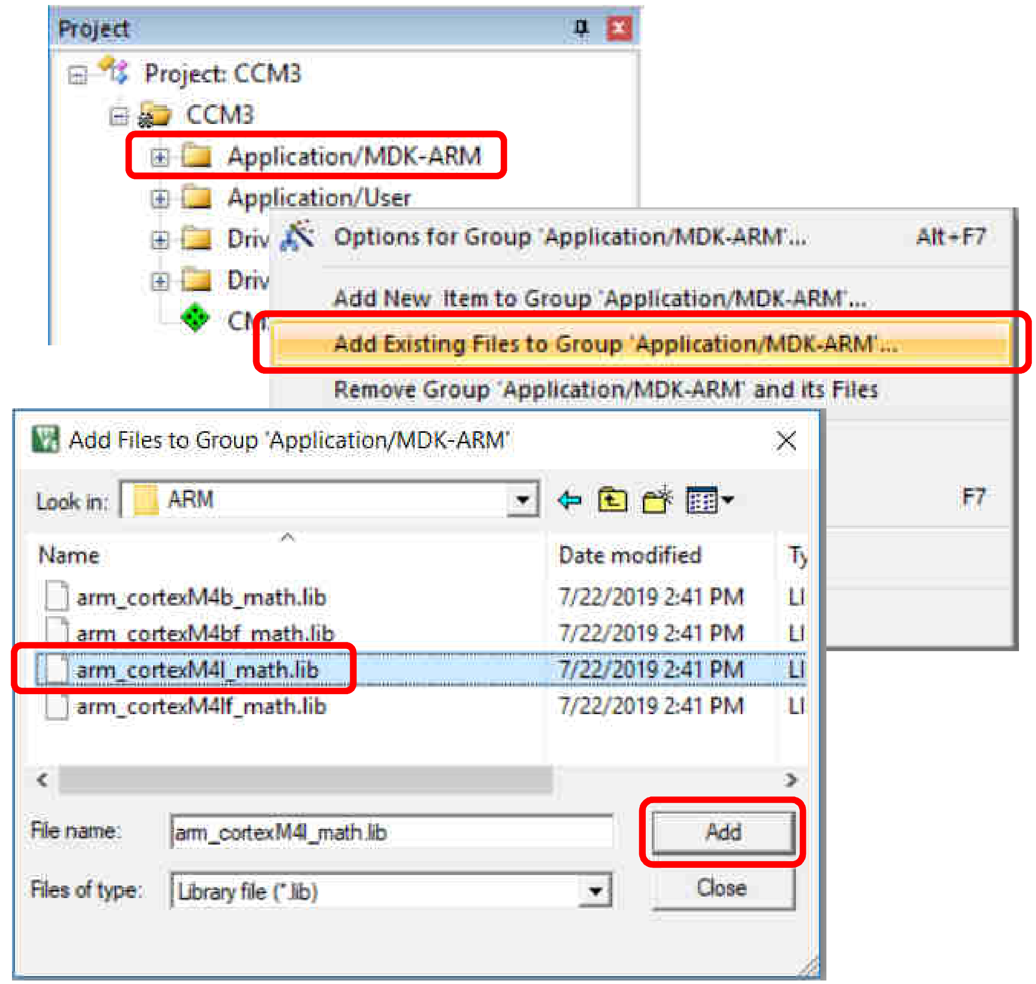
CCM SRAM (library modules)

- Select “C/C++” tab
- Add Define “ARM_MATH_CM4”
- Add library include path
- Click “OK”



CCM SRAM (library modules)

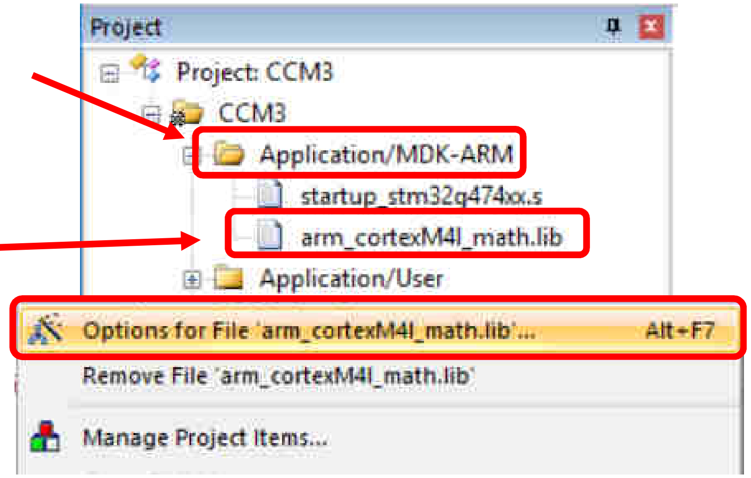
- Add ARM math library
  - Right mouse click on Application/MDK-ARM
  - Add Existing Files to Group
  - Select “arm_cortexM4l_math.lib” file
    - \Drivers\CMSIS\Lib\ARM
- Click “Add” button
- “Close” window

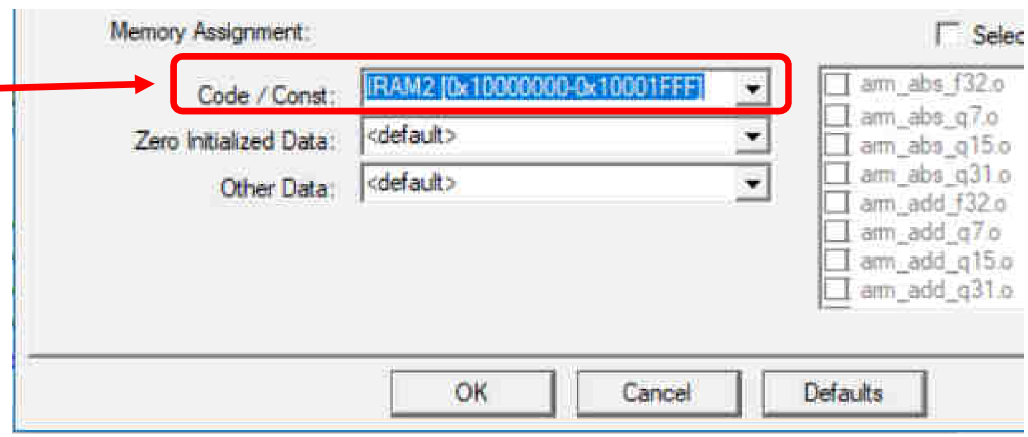


CCM SRAM (library modules)

STM32 G4

- Expand “Application/MDK-ARM” folder
- Right mouse click on arm_cortexM4l_math.lib file
- Select “Options for File ‘arm_cortexM4l_math.lib’…”
- Set Memory Assignment
  - Code = IRAM2 [0x10000000-0x10001FFF]
- Click “OK”

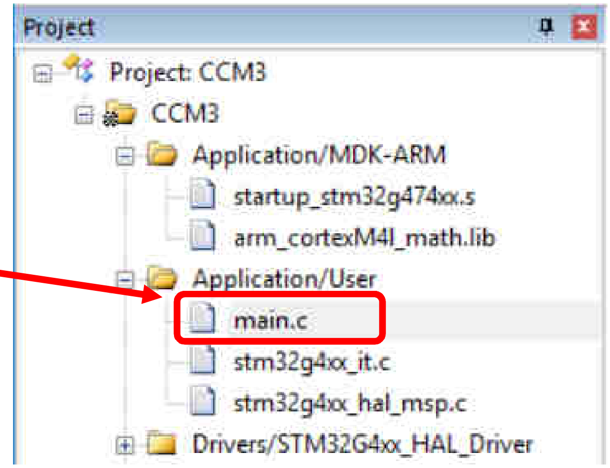




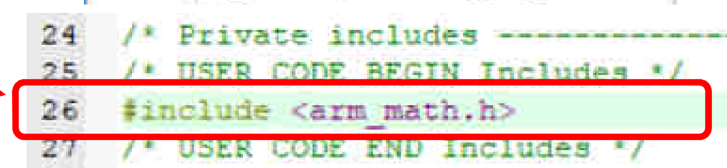
life.augmented

CCM SRAM (library modules)

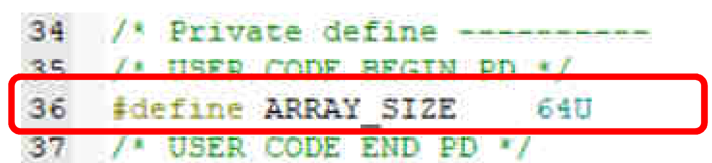
- Expand “Application/User” folder
- Open “main.c” file



- Go to line 26:
  - Add: #include <arm_math.h>



- Go to line 36:
  - Add: #define ARRAY_SIZE 64U



life.augmented

CCM SRAM (library modules)

- Go to line 47:
  - Add: int32_t aAnglesLib array
  - Add: int32_t aResults array
    - Copy from CCM lab code.txt

```
46 /* USER CODE BEGIN PV */
47 /* Array of angles for arm math library Q1.31 forma
48 static int32_t aAnglesLib[ARRAY_SIZE] =
49 {
50   0x00000000, 0x02000000, 0x04000000, 0x06000000,
51   0x08000000, 0x0A000000, 0x0C000000, 0x0E000000,
52   0x10000000, 0x12000000, 0x14000000, 0x16000000,
53   0x18000000, 0x1A000000, 0x1C000000, 0x1E000000,
54   0x20000000, 0x22000000, 0x24000000, 0x26000000,
55   0x28000000, 0x2A000000, 0x2C000000, 0x2E000000,
56   0x30000000, 0x32000000, 0x34000000, 0x36000000,
57   0x38000000, 0x3A000000, 0x3C000000, 0x3E000000,
58   0x40000000, 0x42000000, 0x44000000, 0x46000000,
59   0x48000000, 0x4A000000, 0x4C000000, 0x4E000000,
60   0x50000000, 0x52000000, 0x54000000, 0x56000000,
61   0x58000000, 0x5A000000, 0x5C000000, 0x5E000000,
62   0x60000000, 0x62000000, 0x64000000, 0x66000000,
63   0x68000000, 0x6A000000, 0x6C000000, 0x6E000000,
64   0x70000000, 0x72000000, 0x74000000, 0x76000000,
65   0x78000000, 0x7A000000, 0x7C000000, 0x7E000000
66 };
67
68 static int32_t aResults[ARRAY_SIZE];
```

CCM SRAM (library modules)

- Go to line 114:
  - Add: for loop to calculate “arm_sin_q31”
    - Copy from CCM lab code.txt

```
113   /* USER CODE BEGIN 2 */
114   for (uint32_t j = 0; j < ARRAY_SIZE; j++)
115   {
116     /* Calculate sine */
117     aResults[j] = arm_sin_q31(aAnglesLib[j]);
118   }
119
120   /* USER CODE END 2 */
```

CCM SRAM (library modules)

- “Build” project

  Build (F7)
  Build target files

- “Start Debug” Session

  Start/Stop Debug Session (Ctrl+F5)
  Enter or leave a debug session

- Place breakpoint on call to “arm_sin_q31” (Line 117)

```
114   for (uint32_t j = 0; j < ARRAY_SIZE; j++)
115   {
116     /* Calculate sine */
117     aResults[j] = arm_sin_q31(aAnglesLib[j]);
118   }
```

- “Run” code

  Run (F5)
  Start code execution

- “Step” the code twice

  Step (F11)
  Step one line

- Stops in CCM SRAM region address 0x10000000

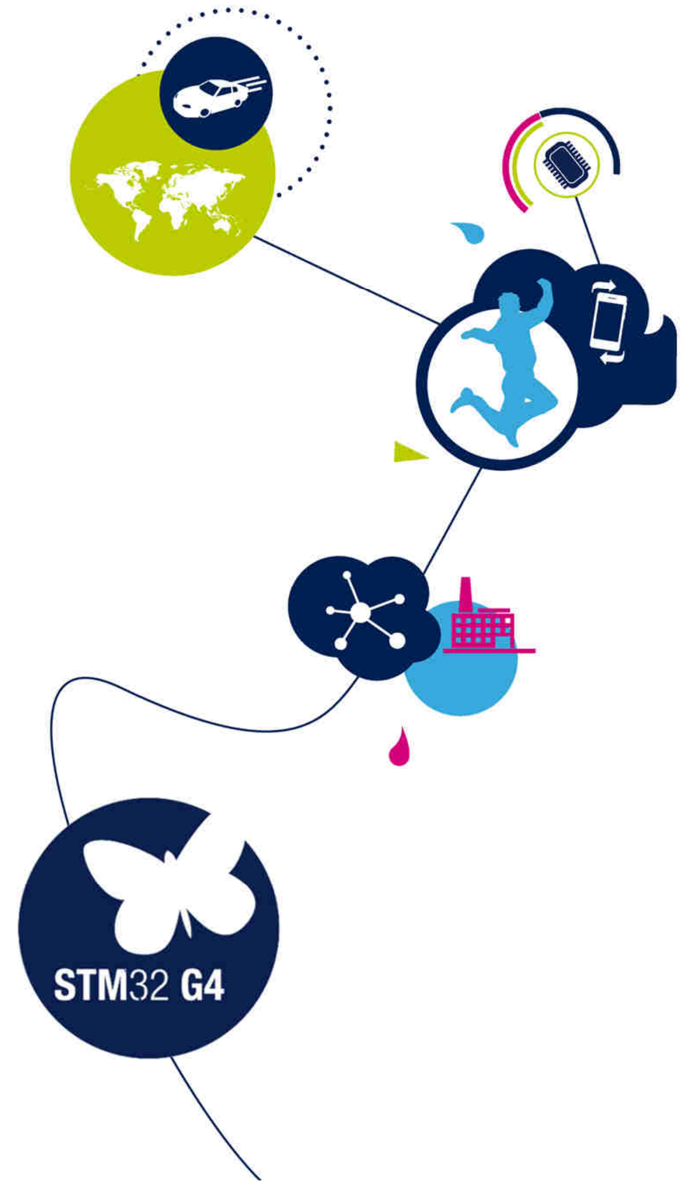
```
                        arm_sin_q31:
0x10000000 B470         PUSH            {r4-r6}
0x10000002 0D81         LSRS            r1,r0,#22
```

CCM SRAM: Additional Notes

- AN4296: Use STM32F3/STM32G4 CCM SRAM with IAR EWARM, Keil MDK-ARM and GNU-based toolchains

STM32G4 mixed-signal MCU workshop

Wrap Up



life.augmented

STM32G4 Workshop Related Survey

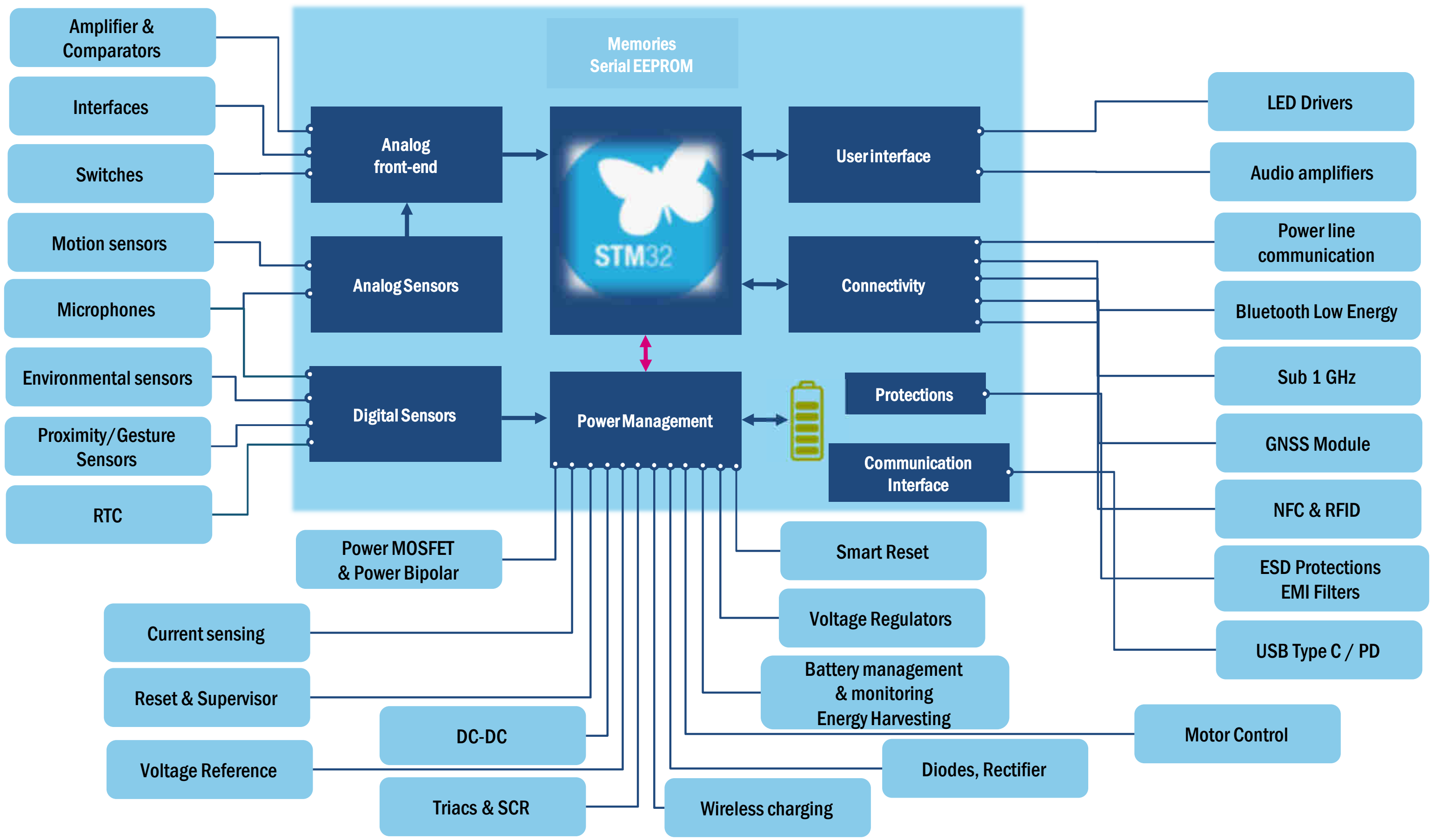
Use your phone to scan the QR code or type the link into your browser.

https://www.surveymonkey.com/r/PC5DHBM

We Greatly Value Your Feedback - Thank you!

STMicroelectronics: one of the broadest Standard Product Portfolios

The largest offerings of companion products for every STM32 Microcontrollers

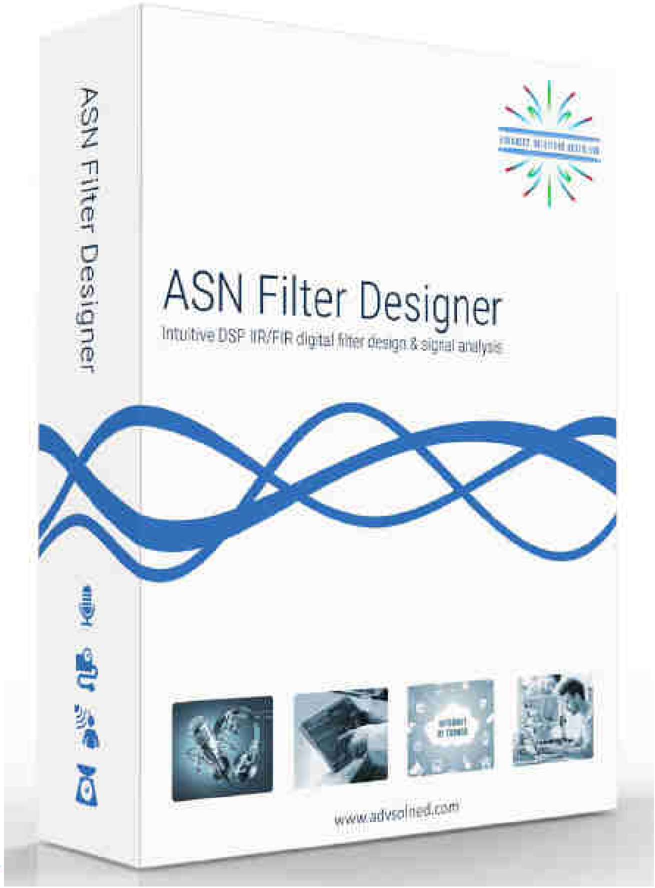


ST Standard Products

- Protection devices
- Power bipolar transistors
- Power MOSFETs
- Amplifiers and Comparators
- Linear voltage regulators
- EEPROM
- Rectifiers
- SCRs and TRIACs
- Interface ICs
- Voltage references
- Reset and supervisor ICs

life.augmented

ASN Filter Designer



Powerful DSP experimentation platform

Design and fine-tune your digital filter within minutes, and then test it in real-time with a variety of test signals (including your own datasets) via the tool's built-in signal analyser.

Real-time design feedback

No need to explicitly define technical specifications, just sketch your requirements and let the tool automatically fill in the exact specifications for you!

Live math scripting

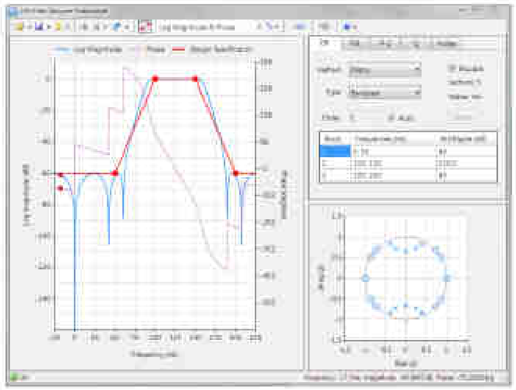
Interactively experiment with specialised transfer functions, pole-zero pairs and other mathematical functions with ASN FilterScript.

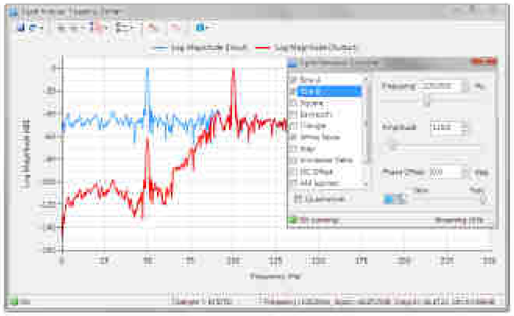
Automatic documentation

Get a detailed specification report of all designed filter blocks for official project documentation, which can be pasted directly into Microsoft Word/Powerpoint.

Deploy to Arm CMSIS-DSP, C/C++, Python, Matlab and more

Easily integrate your designed filter blocks with your other algorithms in Arm CMSIS-DSP, C/C++, Python, Matlab, Octave, Scilab or Labview using provided royalty free software development frameworks.





Download a free demo
http://www.advsolned.com/asn_filter_designer/

STM32 G4

life.augmented

Biricha Digital Power Design Tool

Appendix – Reference Documents

Datasheet and Reference Manual Links

Datasheets

STM32G431xx	STM32G431x6 STM32G431x8 STM32G431xB
STM32G441xB	STM32G441xB
STM32G473xx	STM32G473xB STM32G473xC STM32G473xE
STM32G474xx	STM32G474xB STM32G474xC STM32G474xE
STM32G483xE	STM32G483xE
STM32G484xE	STM32G484xE

Reference Manual

RM0440	STM32G4 Series Reference Manual

Appendix – Reference Documents

Application Note Links

AN2834	How to get the best ADC accuracy
AN4013	STM32 Cross series timer overview
AN4232	Getting started with analog comparators
AN4296	Using CCM SRAM with IAR, Keil, and GNU based tool chains
AN4539	HRTIM Cookbook
AN4899	STM32 GPIO configuration for hardware settings and low power consumption
AN4989	STM32 debug toolbox
AN5305	Digital filter implementation with the FMAC
AN5306	Operational Amplifier usage in STM32G4
AN5310	Guideline for using analog features of the G4 vs. the F3
AN5315	STM32Cube firmware examples for STM32G4
AN5325	Getting started with the CORDIC accelerator
AN5345	High-brightness RGB LED control using the B-G474E-DPOW1 Discovery kit
AN5346	STM32G4 use tips and recommendations

Appendix – Reference Documents

User Manual Links

UM1721	Developing applications on STM32Cube with FatFS
UM1722	Developing applications on STM32Cube with RTOS
UM2454	STM32G4 Seris Safety Manual
UM2492	Getting started with STM32CubeG4 for STM32G4 Series
UM2552	USB Power Delivery
UM2570	Description of STM32G4 HAL and low-layer drivers
UM2573	STM32CubeG4 Nucleo demonstraion firmware
UM2577	Discovery kit with STM32G474RE MCU

STM32G4 Series – Take Away

Analog-rich MCUs for mixed-signal applications

Performance
170MHz Cortex-M4 coupled with 3x accelerators

Rich and Advanced Integrated Analog
ADC, DAC, Op-Amp, Comp.

Safety and security focus

Large portfolio available from NOW!
32..512KB Flash memory
32..128-pin packages

Releasing Your Creativity

/STM32 @ST_World community.st.com stm32g4-online-training

life.augmented

www.st.com/STM32G4